

NAVAL POSTGRADUATE SCHOOL MONTEREY, CALIFORNIA



THESIS

INTERPOLATION TECHNIQUES IN HIGH- RESOLUTION RESIDUE ANTENNA ARCHITECTURES

by

Byeong-Jun Park

September 1996

Thesis Co-Advisors:

David C. Jenn
Phillip E. Pace

Approved for public release; distribution is unlimited.

19970214 004

DTIC QUALITY INSPECTED 1

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.				
1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE September 1996	3. REPORT TYPE AND DATES COVERED Master's Thesis		
4. TITLE AND SUBTITLE INTERPOLATION TECHNIQUES IN HIGH-RESOLUTION RESIDUE ANTENNA ARCHITECTURES		5. FUNDING NUMBERS		
6. AUTHOR(S) Byeong-Jun Park				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000		8. PERFORMING ORGANIZATION REPORT NUMBER		
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSORING/MONITORING AGENCY REPORT NUMBER		
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.				
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.		12b. DISTRIBUTION CODE		
13. ABSTRACT (maximum 200 words) A direction finding antenna based on residue number systems (RNSs) is presented. Special spacing requirements for the array elements are derived and processing of the array output for high speed direction of arrival (DOA) estimates is discussed. The RNS antenna processor encodes the interferometer phase response, which is a sawtooth folding waveform. By design, the phase response of each element pair folds with folding period equal to the chosen modulus. If DOA samples are generated by stepping the emitter direction between -90 and 90 degree in small increments, some samples will fall about the code transition points and result in encoding errors. The resulting DOA estimates from such a realizable system could contain large spikes or "glitches" at these points. These encoding errors in the resolved DOA can be reduced by interpolation. Three primary methods will be discussed to compare the capability of removing the glitches: LSB-Shift Method, Random-LSB Method, and Shift Last Good-Sample Method. A comparison of the performance of the three methods is made on the basis of simulation data.				
14. SUBJECT TERMS Residue Number System, Encoding Error, Interpolation, Glitches, MSB, LSB.		15. NUMBER OF PAGES 117		
		16. PRICE CODE		
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL	

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18 298-10

**INTERPOLATION TECHNIQUES IN HIGH-RESOLUTION RESIDUE
ANTENNA ARCHITECTURES**

Byeong-Jun Park
Captain, ROK Army
B.S., Korea Military Academy, 1990

Submitted in partial fulfillment
of the requirements for the degree of

**MASTER OF SCIENCE
IN
SYSTEMS ENGINEERING**

from the

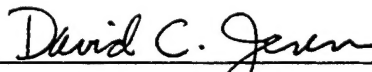
**NAVAL POSTGRADUATE SCHOOL
September 1996**

Author:



Byeong-Jun Park

Approved by:



David C. Jenn, Thesis Co-Advisor



Phillip E. Pace, Thesis Co-Advisor



Frederic H. Levien, Chairman
Electronic Warfare Academic Group

ABSTRACT

A direction finding antenna based on residue number systems (RNSs) is presented. Special spacing requirements for the array elements are derived and processing of the array output for high speed direction of arrival (DOA) estimates is discussed.

The RNS antenna processor encodes the interferometer phase response, which is a sawtooth folding waveform. By design, the phase response of each element pair folds with folding period equal to the chosen modulus.

If DOA samples are generated by stepping the emitter direction between -90 and 90 degree in small increments, some samples will fall about the code transition points and result in encoding errors. The resulting DOA estimates from such a realizable system could contain large spikes or "glitches" at these points. These encoding errors in the resolved DOA can be reduced by interpolation. Three primary methods will be discussed to compare the capability of removing the glitches: LSB-Shift Method, Random-LSB Method, and Shift Last Good-Sample Method. A comparison of the performance of the three methods is made on the basis of simulation data.

TABLE OF CONTENTS

I. INTRODUCTION	1
A. OVERVIEW	1
B. THESIS OUTLINE.....	4
II. DIRECTION FINDING PRINCIPLES AND METHODS	7
A. SPATIAL DIRECTION-FINDING PRINCIPLES	8
B. DIRECTION FINDING TECHNIQUES.....	9
1. Overview	9
2. Amplitude Response	9
3. Phase Difference.....	10
4. Time Delay	12
III. ARRAY BEAMFORMING THEORY	13
A. OVERVIEW	13
B. BASIC SIGNAL AND NOISE MODELS	13
1. Jointly Uncorrelated	19
2. Correlated Coherent	20
C. ARRAY MATRIX DECOMPOSITION FOR MULTIPLE EMITTER DOA ..	20
IV. RESIDUE ANTENNA ARCHITECTURE.....	23
A. OVERVIEW OF RESIDUE NUMBER SYSTEMS.....	23
B. RNS ANTENNA ARCHITECTURE	25
C. RNS ARRAY CORRELATION MATRIX FOR MULTIPLE EMITTERS.....	28

D. COMPUTER SIMULATION OF RNS CODE.....	30
V. ENCODING ERROR CORRECTION	35
A. ENCODING ERRORS.....	35
B. ADDITIONAL COMPARATORS TO ISOLATE ENCODING ERRORS	37
C. INTERPOLATION.....	40
1. LSB Shift Method	42
2. Random LSB Method	51
3. Shift Last Good-Sample Method	56
VI. CONCLUDING REMARKS	61
APPENDIX MATLAB CODE.....	63
LIST OF REFERENCES	99
INITIAL DISTRIBUTION LIST	101

LIST OF FIGURES

1. DF Array with RNS Processing	3
2. DF Spatial Coordinate System	8
3. Azimuth Plane Response Patterns	10
4. Phase-difference DF Parameters	11
5. Differential Time-of-arrival Parameters	12
6. Geometry of an N Element Linear Array with M Incident Plane Waves.....	14
7. Conventional Equally Spaced Array with Six Elements when the Emitter Directions are 20, 50, 90, 130, and 160	22
8. Conventional Processing Scheme for a Caratheodory-array with Four Elements when the Emitter Directions are 20, 50, 90, 130, and 160	22
9. RNS Folding Waveforms and Output Code for the Moduli (3,4,5)	24
10. RNS Antenna Architecture	26
11. RNS Antenna Spacing for the (3,4,5) Array.....	28
12. RNS Array Response for Multiple Emitters when the Emitter Directions are 50, 90, and 130.....	30
13. Phase Response as a Function of $\sin\theta$ for the Moduli (3,4,5).....	31
14. Phase Response as a Function of the Input Direction of Arrival (θ) the Moduli (3,4,5)	for 32
15. Resolved Direction of Arrival in Function of the Input Direction of Arrival when DOA Increment is 0.3°	34
16. RNS-to-Decimal Logic Block and Glitch.....	36
17. Glitches and Resolved Direction of Arrival	37
18. Original Threshold Level in Modulus 3	38
19. Additional Threshold Levels and RNS Output Code.....	39
20. The Procedure of DOA Decoder.....	40
21. Parity Circuit and Error Correction Block.....	43

22. LSB-Shift Method in Case of 1/5 Error Samples	44
23. LSB-Shift Method in Case of 2/5 Error Samples	45
24. LSB-Shift Method in Case of 3/5 Error Samples	46
25. LSB-Shift Method in Case of 4/5 Error Sample Case	47
26. LSB-Shift Interpolation Results, 1% of LSA	48
27. LSB-Shift Interpolation Results, 2% of LSA	49
28. LSB-Shift Interpolation Results, 3% of LSA	49
29. LSB-Shift Interpolation Results, 4% of LSA	50
30. LSB-Shift Interpolation Results, 5% of LSA	50
31. Number of Glitches vs LSA Percentage After LSB-Shift Method	51
32. Procedure of Random LSB Method	52
33. Random-LSB Method Result, 1% of LSA	53
34. Random-LSB Method Result, 2% of LSA	53
35. Random-LSB Method Result, 3% of LSA	54
36. Random-LSB Method Result, 4% of LSA	54
37. Random-LSB Method Result, 5% of LSA	55
38. Number of Glitches vs LSA Percentage after Random LSB Method	55
39. Procedure of Shift Last Good-Sample Method	57
40. Shift Last Good-Sample Method Result, 1% of LSA	58
41. Shift Last Good-Sample Method Result, 2% of LSA	58
42. Shift Last Good-Sample Method Result, 3% of LSA	59
43. Shift Last Good-Sample Method Result, 4% of LSA	59
44. Shift Last Good-Sample Method Result, 5% of LSA	60
45. Number of Glitches vs LSA Percentage After Shift Last Good-Sample Method	60
46. Comparison of the Three Interpolation Method Results	62

LIST OF TABLES

1. RNS Encoding Procedure for the Moduli (3,4,5).....	25
2. Threshold Levels for the Moduli (3,4,5).....	31
3. RNS Code and Equivalent Resolved Direction.....	33
4. Encoding Error in the RNS Output Code	35
5. New threshold Levels for the Moduli (3,4,5)	37
6. LSA Percentage and Six-Additional Threshold Values	39
7. RNS code and Six-bit Binary Code in Moduli (3,4,5)	41
8. Encoding Error in Six-bit Binary Code	42

ACKNOWLEDGMENT

I would like to thank Professor David C. Jenn for his direction, insight and advice during the thesis process. I would also like to thank Professor Phillip E. Pace, for his encouragement and capable guidance. I would also like to thank others who made this work possible, Microwave Laboratory assistant Bob Vitale and my fellow students. Without their help my effort would never have been successful. A special mention goes to the Korean Army which has given me the opportunity and support for graduate studies at the Naval Postgraduate School. But most of all, I'd like to express thanks to God for making this long tour successful.

This research was supported by the Space and Naval Warfare Systems Command PMW-163 and was essential to make this study possible.

I. INTRODUCTION

A. OVERVIEW

Direction finding (DF) is the area of electronic-warfare support (ES) whose objective is to find the direction of arrival (DOA) of a targeted transmitter. ESM performs search, interception, location, and identification of hostile radiations. DF systems passively examine the spectrum use by hostile emitters and process the signals to obtain enemy DOA information.

A general assumption in most DF systems is that the received field exhibits far-field plane-wave behavior with linear polarization. In reality, incident fields are often non-planar with phase-front distortion caused by multipath and scattering. Typical DF system techniques assume that only a single signal source is received. Because of multipath ionospheric propagation (skywave), and wave scattering by objects close to the DF site or co-channel transmitting stations, signals via multiple paths and sources are often received at DF sites. A DF system designed to handle one source at a time can work in a multiple source environment if other signals are well separated in frequency, time, or direction of arrival.

In practice, DF stations often receive signals from several sources in the same frequency range at the same time, and from many different angles of arrival [Ref. 1]. To resolve multicomponent wavefields, modern techniques of spectral estimation methods are used. These methods include eigenanalysis and linear prediction.

Modern direction finding systems fall into three main categories: interferometer-techniques, scanning beam, and simultaneous-multiple beam. Interferometer systems based on phase-comparison techniques, have the advantage of a fast response, but generally use wide coverage antennas which result in low sensitivity. In addition, they require relatively complex microwave circuitry that must maintain a precise phase match over a wide frequency band under extreme environmental conditions. When high accuracy is required, wide baseline interferometers are used with ambiguity resolving circuitry. The mechanically

scanning beam requires only a single antenna. However, this technique has the disadvantage of a low probability of intercept. The simultaneous multiple-beam system uses an antenna or several antennas, forming a number of simultaneous beams, thereby retaining the high sensitivity of the scanning beam approach while providing fast response. However, it requires many parallel receiving channels, each with full frequency coverage.

This thesis is concerned with a technique for radiating element placement and signal processing based on the residue number systems (RNS). In principle, the direction of arrival can vary in azimuth (0-360 degrees) and elevation (0-90 degrees). However, this paper deals exclusively with the azimuth angle of arrival for a linear array. In addition, an underlying assumption is that the receiver is far enough from the transmitter so that the arriving wavefront is essentially planar.

The basic concept of residue arrays¹ was presented in a previous thesis by Rodrigues [Ref. 2]. A simple block diagram of such a system is shown in Figure 1. It is essentially an interferometer type DF system consisting of $N-1$ pairs of elements. Each pair includes element 1 so that there are $N-1$ pairs if the total number of elements is N . The number of pairs is determined by the number of moduli employed in processing the signal. It will be seen that the number of moduli controls the angle resolution, the element spacings, and the number of comparators required in the processor. The phase difference between each pair of elements is obtained using a phase detector. In principle, phase or amplitude comparison can be used, but Rodrigues has shown that the phase is less sensitive to variations in the DOA and element factors. The phase differences are quantized and encoded using the RNS [Ref. 3]. The encoded data from all pairs is combined to obtain a DOA estimate.

This DF method has several attractive features. First, arbitrarily fine unambiguous angle resolution can be achieved with as few as three elements. Secondly, the method is fast, providing a DOA estimate limited only by the RNS processor speed. The processor

¹ This term will be used throughout the thesis to refer to arrays with element locations or encoding algorithms based on residue or symmetrical number systems.

employed here is based on parallel digitization of the $N-1$ phase signals, and therefore nearly instantaneous DOA estimates are obtained.

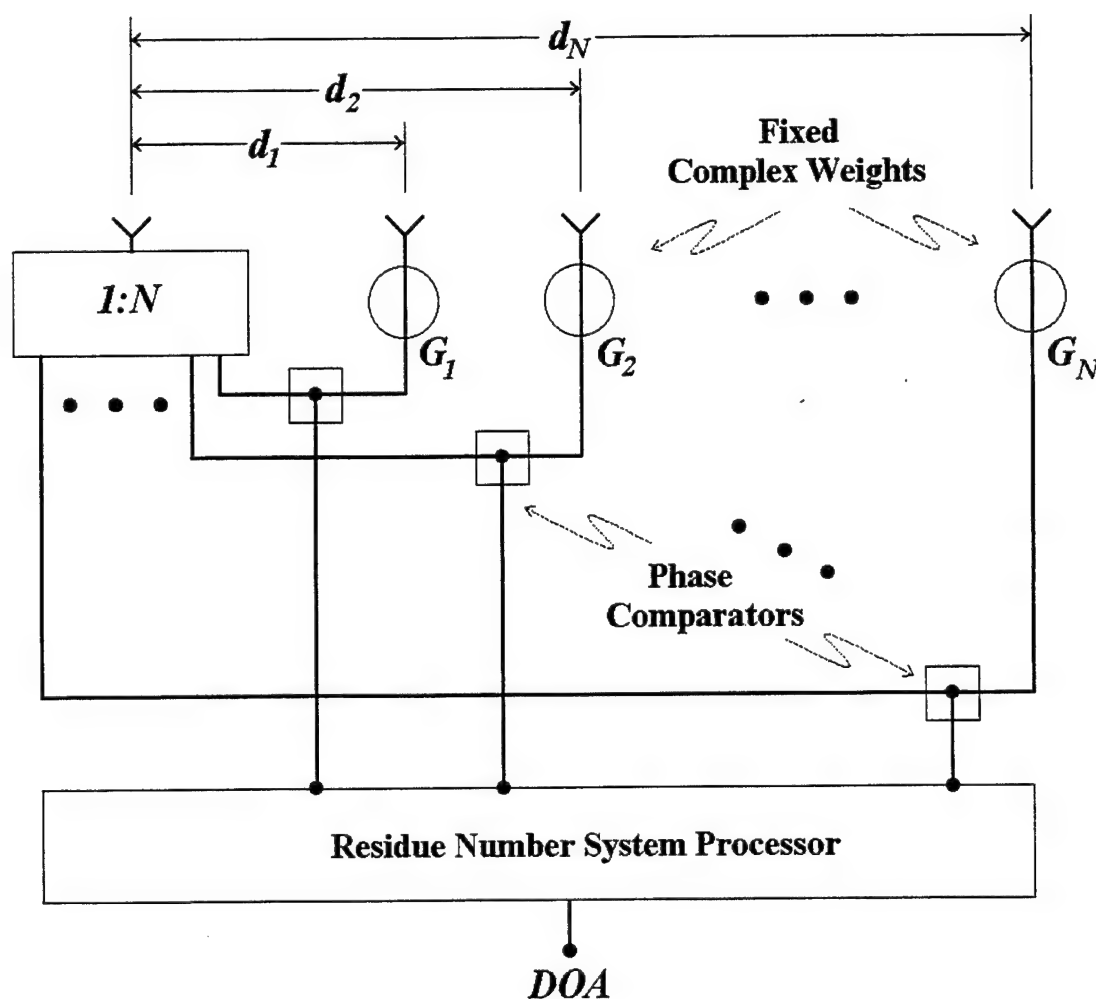


Figure 1. DF Array with RNS Processing

The RNS DF method has several potential problems. They are primarily performance limitations imposed by imperfections in the system hardware. Several of these occur in the array beamforming network:

1. Mutual coupling variations due to the non-periodic separation of elements.
2. Imbalance in the weights and transmission lines.

3. Low accuracy of the phase comparators due to low signal levels.

Other errors are introduced in the RNS processing:

1. Encoding errors that arise from variations in the comparator thresholds.
2. Increase in quantization error due to frequency deviations.

The most serious problem is the encoding errors which, because of the non-sequential binary encoder outputs, can cause extremely large errors in the DOA estimate. In fact, the net effect of all of the errors listed above can be reduced to an equivalent encoding error.

This thesis examines three interpolation schemes that can be used to reduce the encoding errors which result from quantization in the RNS signal processing. Simulation data shows, for example, that small errors in the comparator thresholds can result in large DOA errors. Interpolation methods can be used to reduce the occurrence and magnitude of the errors.

Three types of interpolation are investigated: (1) a shifted least significant bit (LSB) method, (2) a randomized LSB method, and (3) a shift last good sample method. The effectiveness of these three approaches is examined using computer simulation.

B. THESIS OUTLINE

Chapter II summarizes common direction finding techniques such as amplitude response, phase difference, and time delay methods. Chapter III covers array beamforming and the mathematical development for a linear array antenna. Also, direction finding for multiple emitter DOAs is introduced by using the array correlation matrix. Chapter IV is devoted to the process of encoding the phase response with the RNS. First the RNS encoding procedure is briefly introduced. The RNS antenna architecture is then derived with simulation results shown to demonstrate the array performance. Next, we examine encoding the phase response with the RNS. The RNS encoding procedure is briefly described and simulation results are presented. In Chapter V, the encoding error correction algorithms are derived. Three primary methods are investigated to remove the quantizing

errors or "glitches." A comparison of the performance of the three methods is made on the basis of simulation data. Chapter VI presents a summary, some concluding remarks, and recommendations for future work.

II. DIRECTION FINDING PRINCIPLES AND METHODS

As early as 1899 it was recognized that electromagnetic waves could be aimed or beamed in a specific direction by the use of antennas such as loops or spaced dipoles. For receiving purposes this meant that the direction of arrival of a radio wave could be determined by moving or steering a directive antenna beam in azimuth until a maximum signal was obtained. The beam could be steered by physically moving an antenna or by appropriately summing the outputs of the individual elements of an array.

The science of direction finding has been the subject of much study since the early 1900's. A major improvement over early DF systems came with the development of the Adcock DF system employing four or more spaced elements with an overall aperture of one-half wavelength or less. The Adcock system achieves an instrumental accuracy of about 5 degrees.

Since the end of World War II, there has been a trend toward larger aperture systems. For example, Doppler DF systems, in which frequency modulation (FM) by a sinusoid is induced on the received signal by a rotating antenna or its equivalent, usually range in size from one-half to five wavelengths and are capable of accuracies of 2 degrees or less [Ref. 4].

The Wullenweber DF system, which was developed in Germany in the early 1940's, consists of a large circular array on the order of ten wavelengths in diameter [Ref. 5]. At any instant of time only one segment of the array is used, and with proper phasing of the elements in use, this segment is made equivalent to a planar array. By commutation the planar array is in essence caused to move in a circle. An accuracy of 0.5 degree or less is possible with this system.

At microwave frequencies, antennas having aperture sizes of several wavelengths and with extremely directive patterns can be physically rotated to give accurate DOA information. An alternate method which is very accurate for large values of signal-to-noise

ratio (SNR) is the amplitude-sensing monopulse technique in which two identical directive antennas are placed some distance apart.

A. SPATIAL DIRECTION-FINDING PRINCIPLES

The main function of a direction finder is to find out the direction of arrival of an incident electromagnetic wave relative to the coordinates of the direction finding site. Figure 2 shows a representative direction-finding spatial coordinate system with the DF located at the origin coordinate point. The direction of arrival is specified by the azimuth angle, measured from the y-axis, and the elevation angle measured from the z-axis.

General direction-finding principles assume that the electromagnetic field incident on the DF exhibits a far-field planar wave structure with linear polarization. As depicted in Figure 2, the radial E-field is zero. The E-field, and the H-field are in space quadrature [Ref. 6].

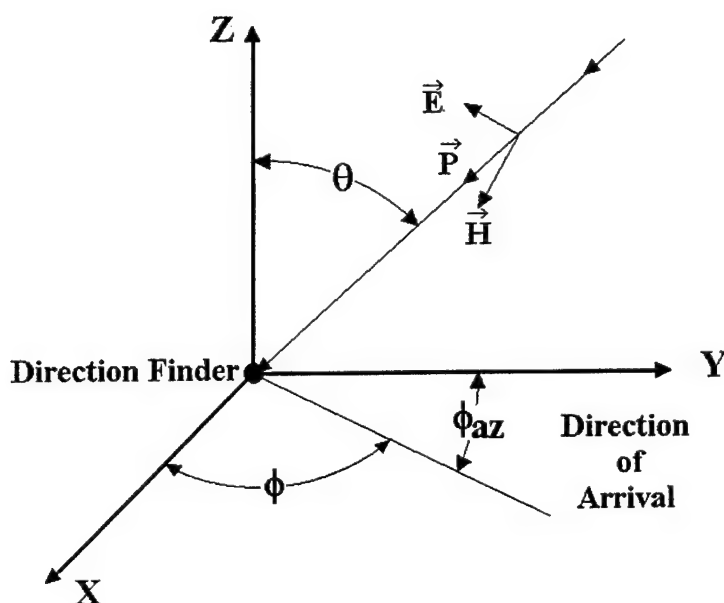


Figure 2. DF Spatial Coordinate System

The direction of propagation is indicated by Poynting's vector, \vec{P} . In practice, the incident electromagnetic field is usually nonplanar with phase-front distortion, created by multipath

and scattering, and depolarization produced by nonuniform ionospheric propagation effects. A full-capacity generic DF should measure direction of arrival in three-dimensional space. However, many operational situations require only measurement of the directional component in the azimuth plane.

The measured angle of arrival, ϕ_{az} , is usually called a bearing angle or line of bearing (LOB). Before examining residue arrays, the capabilities of several conventional DF methods are presented.

B. DIRECTION FINDING TECHNIQUES

1. Overview

Common direction-finders determine direction-of-arrival information by one of three measurement methods [Ref. 6]:

1. Amplitude response,
2. Phase difference,
3. Time delay.

The angle of arrival is converted into a voltage analog using amplitude response, phase difference, or time delay. Basic conversion techniques are presented in the following sections.

2. Amplitude Response

Two types of amplitude response methods are used to obtain DF information: direct and comparative. For direct amplitude response, the directional properties of elemental H-field loops and E-field dipoles provide response minima as they are rotated in azimuth. Bearing data are acquired at these minima response orientations.

Elemental loops and dipoles exhibit figure-eight responses as depicted in Figure 3 for the azimuth plane. Response maxima are broad but the minima (nulls) are sharp. The amplitude response derivative is greatest around the null response position. Therefore, a dynamic measure of null position provides the best bearing estimates.

Sequential amplitude measurements are made as the directive pattern is rotated in azimuth, and the measured bearing is indexed to the azimuth at which some predetermined amplitude-null property occurs.

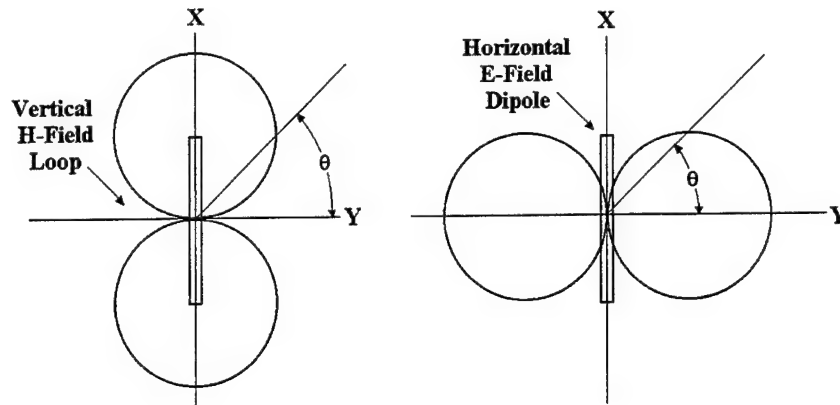


Figure 3. Azimuth Plane Response Patterns

The horizontal E-field dipole is rarely used for DF purposes, primarily because most signals of interest are vertically polarized and of orthogonal polarization to the horizontal dipole [Ref. 6].

For comparative amplitude DF, multiple antennas are used to obtain orthogonal patterns. Bearing information is obtained from the ratio of the amplitudes of the orthogonal patterns. Comparative amplitude DF techniques provide instantaneous bearing information.

3. Phase Difference

Bearing measurements using a phase difference require at least two separated antennas. A plane wave, arriving at an angle other than the normal to the baseline between the two elements, arrives at one element earlier than the other. The time lag between the antennas produces a RF phase delay or a differential RF phase between the antenna outputs.

Figure 4 illustrates the basic phase-delay technique. An incident plane wave arrives at an incident angle θ at antenna 1 inducing a voltage $V_1 \exp(j\omega t)$. After propagating the

distance $d \sin \theta$, the incident plane wave induces a voltage $V_2 \exp(j\omega t - \Phi)$ in antenna 2, where the phase delay given by

$$\Phi = (2\pi d / \lambda) \sin \theta . \quad (1)$$

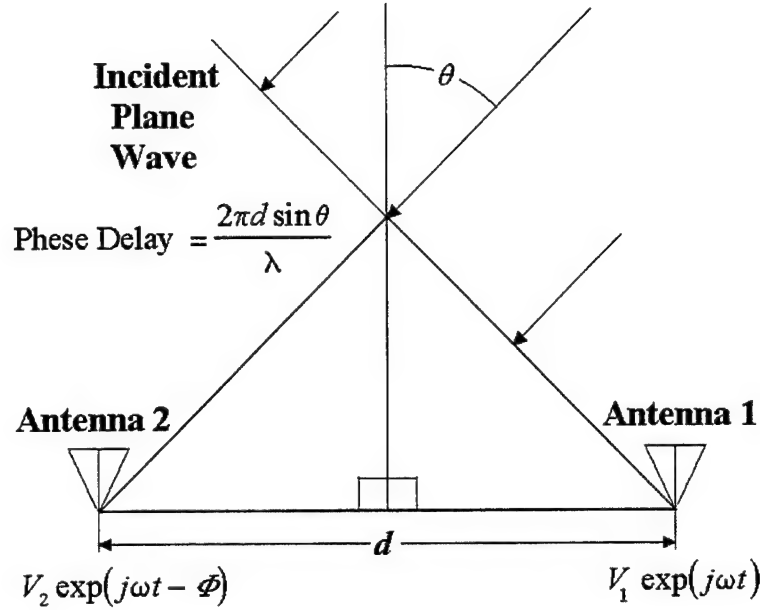


Figure 4. Phase-difference DF Parameters

Therefore the bearing angle is encoded as a function of phase delay. For the phase difference technique, the bearing angle is computed by using Eq. 1, where phase difference is measured and baseline distance d and wavelength λ are known. This phase-delay technique can experience phase and bearing ambiguities. If $d > \lambda/2$ phase ambiguities exist; however, bearing ambiguities are resolved by using auxiliary antennas to establish known phase references [Ref. 6].

A variant of the phase measurement method is Doppler direction finding. The received frequency from a moving antenna experiences a Doppler shift. If the antenna moves along a radial to the emitter position, the Doppler shift is maximized. No Doppler shift occurs if the antenna movement is tangential to the direction of propagation. These

maxima and minima Doppler shifts may be used to measure the bearing of the incident signal.

4. Time Delay

An electromagnetic wave propagating between two separated antennas experiences a time delay. This time differential of arrival (TDOA) contains bearing angle information. In Figure 5, the differential time delay, Δt , between antennas 1 and 2 is given by

$$\Delta t = \frac{d}{c} \cos \theta \quad (2)$$

where:

d = the distance between antennas 1 and 2 (meters);

c = velocity of light;

θ = bearing angle (degrees).

Historically, TDOA has been used for pulsed emissions such as radar. Also, relatively long, multiple-wavelength baselines are commonly used [Ref. 6].

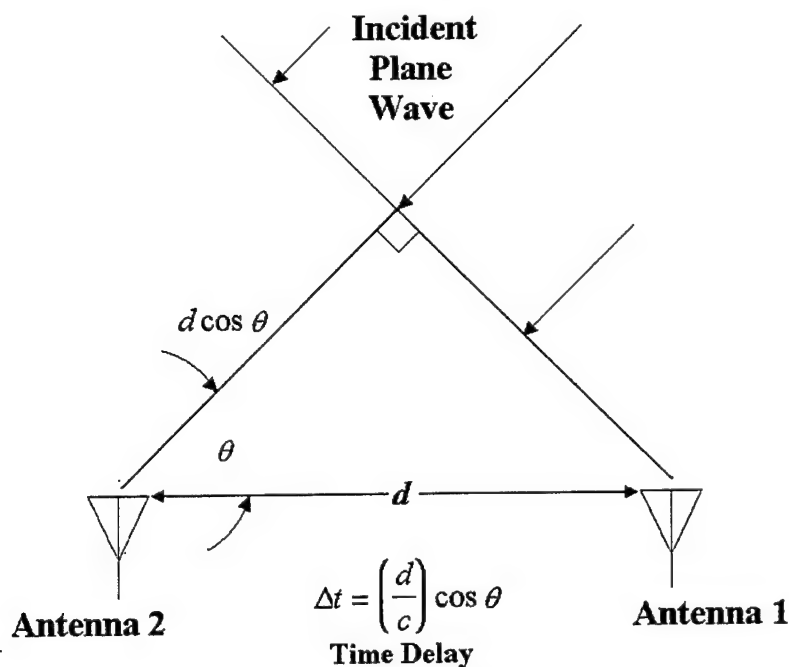


Figure 5. Differential Time-of-arrival Parameters

III. ARRAY BEAMFORMING THEORY

A. OVERVIEW

Direction finding systems consists of an antenna (or collection of antennas), beamforming networks, detectors, and signal processing equipment. It operates by receiving and then analyzing the nature of the emitter signal. In noise-limited situations, the source signals may consist of a self-product and additive receiver noise. In more complicated situations, it may also contain a noise component due to backscatter from the surrounding environment (clutter). A further source of complication may be the presence of interferences produced by jammers beamed at the DF site from directions unknown to operators at the site. In the following analysis, unless otherwise noted, these complications are neglected. Emitter signals are modeled as time-harmonic plane waves, narrowband assumptions are made, and noise is assumed to be white Gaussian. Before examining residue arrays, the capabilities of several conventional DF methods are presented.

B. BASIC SIGNAL AND NOISE MODELS

Figure 6 shows a uniformly spaced linear array consisting of N elements, and with M plane waves arriving at the array from distinct directions. We assume that all these plane waves are narrowband with the same carrier frequency f_c . The noiseless signal produced at the n th element of the array by the wave from source m , denoted as the m th plane wave, is as follows:

$$s(n, m, t) = A_m \cos \left[2\pi f_c t + \left(n - \frac{N+1}{2} \right) \vec{k}_m \bullet \hat{z} \right] \quad (3)$$

where:

$n = 1, 2, \dots, N$ (element index)

$m = 1, 2, \dots, M$ (emitter index)

f_c = carrier frequency

\vec{k}_m = vector wavenumber of the m th incident plane wave = $\frac{2\pi}{\lambda} \hat{k}_m$

\hat{z} = unit vector along the line of the array

t = time

The parameter A_m denotes the amplitude of the signal $s(n, m, t)$.

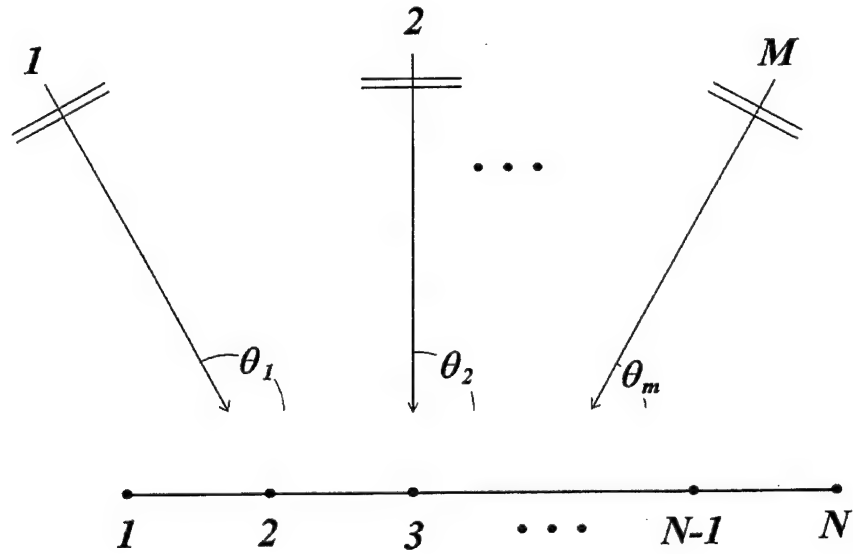


Figure 6. Geometry of an N Element Linear Array with M Incident Plane Waves

Let θ_m denote the angle of arrival of the m th plane wave, measured with respect to the normal to the array, as illustrated in Figure 6. Then we may express the dot product of the two vectors \vec{k}_m and \hat{z} as follows [Ref. 7]:

$$\vec{k}_m \cdot \hat{z} = \frac{2\pi d}{\lambda} \sin \theta_m \quad (4)$$

where d is the interelement spacing and λ is the received radar wavelength. Accordingly, we may rewrite Eq. (3) as:

$$s(n, m, t) = A_m \cos \left[2\pi f_c t + 2\pi \frac{d}{\lambda} \left(n - \frac{N+1}{2} \right) \sin \theta_m \right]. \quad (5)$$

We may simplify the representation of the signal $s(n, m, t)$ by adopting the complex notation for narrowband signals. In particular, we may represent the signal $s(n, m, t)$ by its complex amplitude, defined by:

$$s(n, m) = A_m \exp \left[2\pi j \frac{d}{\lambda} \left(n - \frac{N+1}{2} \right) \sin \theta_m \right]. \quad (6)$$

Now define the electrical phase angle from element to element along the array as:

$$\phi_m = 2\pi \frac{d}{\lambda} \sin \theta_m. \quad (7)$$

Also, let a_m denote the complex amplitude of the signal $s(n, m, t)$ measured at the center of the array; that is:

$$a_m = s \left(\frac{N+1}{2}, m \right). \quad (8)$$

Note that if the number of elements N is odd, the center of the array represented by $n = \frac{N+1}{2}$ corresponds to the center element. On the other hand, when N is even, the center of the array is a fictitious point. In any event, we may use Eq. 7 to rewrite Eq. 6 in the simplified form

$$s(n, m) = a_m \exp \left[j \left(n - \frac{N+1}{2} \right) \phi_m \right]. \quad (9)$$

The signal measured at the output of any element in the array differs from the signal actually received by that element by an amount attributed to noise. In practice, this noise is

modeled as a white ergodic random process. To limit the effects of the noise, it is customary to filter the output of each element in the array to the same narrow band of frequencies occupied by the actual received signal. As a result of this operation, we have an observed signal, which for each element of the array, consists of the actual received signal plus a narrowband noise. For element n we may describe this narrowband noise by

$$w(n, t) = B_n \cos(2\pi f_c t + \beta_n) \quad (10)$$

where the amplitude B_n is a Rayleigh distributed random variable and the phase β_n is uniformly distributed random variable over the range $(0, 2\pi)$. Representing the narrowband noise $w(n, t)$ by its complex amplitude, $w(n)$, we have:

$$w(n) = B_n \exp(j\beta_n) . \quad (11)$$

This is a complex-valued random variable that is Gaussian-distributed with a mean that is typically zero.

Equation 9 defines the complex amplitude of the signal received by element n due to the m th plane wave. Summing the contributions produced by a set of M plane waves, and then adding to the result the complex amplitude of the narrowband noise, we may express the complex amplitude of the observed signal as:

$$\begin{aligned} x(n) &= s(n) + w(n) \\ &= \sum_{m=1}^M s(n, m) + w(n) \\ &= \sum_{m=1}^M a_m \exp\left[j\left(n - \frac{N+1}{2}\right)\phi_m\right] + w(n), \quad n = 1, 2, \dots, N . \end{aligned} \quad (12)$$

We may put Eq. 12 into a more compact form by using matrix notation. First introduce the following definitions:

1. An $N \times I$ vector

$$\mathbf{X} = \begin{bmatrix} x(1) \\ x(2) \\ \vdots \\ x(N) \end{bmatrix} \quad (13)$$

called the observed signal vector.

2. An $N \times 1$ vector

$$\mathbf{S} = \begin{bmatrix} s(1) \\ s(2) \\ \vdots \\ s(N) \end{bmatrix} = \begin{bmatrix} \sum_{m=1}^M s(1, m) \\ \sum_{m=1}^M s(2, m) \\ \vdots \\ \sum_{m=1}^M s(N, m) \end{bmatrix} \quad (14)$$

called the received signal vector.

3. An $M \times 1$ vector

$$\mathbf{A} = \begin{bmatrix} a(1) \\ a(2) \\ \vdots \\ a(M) \end{bmatrix} \quad (15)$$

called the signal-in-space vector.

4. An $N \times M$ matrix

$$\mathbf{B} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \exp(jkd \cos \theta_1) & \exp(jkd \cos \theta_2) & \cdots & \exp(jkd \cos \theta_M) \\ \vdots & \vdots & \vdots & \vdots \\ \exp(jk(N-1)d \cos \theta_1) & \exp(jk(N-1)d \cos \theta_2) & \cdots & \exp(jk(N-1)d \cos \theta_M) \end{bmatrix} \quad (16)$$

of propagation delays called the direction matrix.

5. An $N \times 1$ vector

$$\mathbf{W} = \begin{bmatrix} w(1) \\ w(2) \\ \vdots \\ w(N) \end{bmatrix} \quad (17)$$

called the noise vector.

Then, using these definitions, we may rewrite Eq. 12 in the form

$$\mathbf{X} = \mathbf{S} + \mathbf{W} \quad (18)$$

or, equivalently,

$$\mathbf{X} = \mathbf{BA} + \mathbf{W} \quad (19)$$

where

$$\mathbf{S} = \mathbf{BA} . \quad (20)$$

The observed signal vector \mathbf{X} represents a snapshot of data corresponding to a particular instant of time. Ordinarily, several independent measurements are made, so that the data available for processing may be expressed in the form:

$$\mathbf{X}(i) = \mathbf{BA}(i) + \mathbf{W}(i), \quad i = 1, 2, \dots, I \quad (21)$$

where I is the total number of snapshots taken.

Based on such a set of data, we may use temporal averaging to improve the estimation of the angles of arrival of the plane waves incident upon the array. The signal-in-space vector $\mathbf{A}(i)$ is modeled as a stochastic process because the behavior of the sources responsible for its generation is, in general, unpredictable.

The spatial correlation matrix of the observed signal vector \mathbf{X} is defined by

$$\mathbf{R} = E[\mathbf{X}^*(i)\mathbf{X}^T(i)] \quad (22)$$

where E is the expectation operator, the asterisk denotes complex conjugation, and the superscript T denotes transposition. Substituting Eq. 21 in Eq. 22, and recognizing that the signal-in-space vector \mathbf{A} and the noise vector \mathbf{W} are statistically independent, we get

$$\begin{aligned}\mathbf{R} &= \mathbf{E}[\mathbf{B}^* \mathbf{A} \mathbf{A}^T \mathbf{B}^T] + \mathbf{E}[\mathbf{W}^* \mathbf{W}^T] \\ &= \mathbf{B}^* \mathbf{R}_a \mathbf{B}^T + \mathbf{R}_w\end{aligned}\quad (23)$$

where:

$$\begin{aligned}\mathbf{R}_a &= \text{spatial correlation matrix of the signal - in - space vector } \mathbf{A} \\ &= \mathbf{E}[\mathbf{A}^* \mathbf{A}^T] \\ \mathbf{R}_w &= \text{spatial correlation matrix of the noise vector } \mathbf{W} \\ &= \mathbf{E}[\mathbf{W}^* \mathbf{W}^T] .\end{aligned}$$

The matrix product

$$\mathbf{R}_s = \mathbf{B}^* \mathbf{R}_a \mathbf{B}^T \quad (24)$$

represents the spatial correlation matrix of the signal component of the signal vector \mathbf{S} .

With the noise vector \mathbf{W} assumed white noise, we have

$$\mathbf{R}_w = \sigma_w^2 \mathbf{I} \quad (25)$$

where σ_w^2 is the variance of the elemental noise $w(n)$ for all n , and \mathbf{I} represents the identity matrix. Depending on the structure of the spatial correlation matrix \mathbf{R}_a , we may distinguish two different cases:

1. Jointly Uncorrelated

The sources responsible for the signal-in-space vector \mathbf{A} are jointly uncorrelated or noncoherent so that the off-diagonal elements of the corresponding spatial correlation matrix \mathbf{R}_a are all zero. Then we find that the spatial correlation matrix of the received signal vector, $\mathbf{R}_s = \mathbf{B}^* \mathbf{R}_a \mathbf{B}^T$ is a Toeplitz matrix. A square matrix is Toeplitz if all the elements on its main diagonal are equal, and likewise for the elements on any other diagonal parallel to the main diagonal. The received signal vector \mathbf{S} is spatially stationary if its spatial correlation matrix is Toeplitz, or vice versa. Clearly, if the spatial correlation matrix \mathbf{R}_s is Toeplitz, the spatial correlation matrix of the observed signal vector, \mathbf{R} is likewise Toeplitz.

2. Correlated Coherent

The sources responsible for the received signal vector \mathbf{S} are correlated coherent. The result is that the spatial correlation matrix \mathbf{R}_s is *non-Toeplitz*, and the received signal vector \mathbf{S} is said to be spatially nonstationary. Clearly, If \mathbf{R} is non-Toeplitz, the spatial correlation matrix, \mathbf{R} , pertaining to the observed signal vector, is also non-Toeplitz.

C. ARRAY MATRIX DECOMPOSITION FOR MULTIPLE EMITTER DOA

For the location of the directions of arrival, the autocorrelation matrix takes the form of a Toeplitz matrix.

$$\begin{aligned}\mathbf{X} &= \mathbf{B}\mathbf{A} + \mathbf{W} \\ \mathbf{A} &= [a(1), a(2), \dots, a(M)]^T \\ \mathbf{W} &= [w(1), w(2), \dots, w(N)]^T\end{aligned}\tag{26}$$

Vector \mathbf{X} is explained in Eq. 13 and vector \mathbf{B} is defined in Eq. 16; it contains the propagation delays, that is $\mathbf{B} = [\mathbf{U}_1, \mathbf{U}_2, \dots, \mathbf{U}_M]$ where $\mathbf{U}_m = \mathbf{U}(\theta_m)$, and, for equally spaced elements,

$$\mathbf{U}(\theta) = \begin{bmatrix} e^{jkd_1 \cos \theta} \\ \vdots \\ e^{jkd_N \cos \theta} \end{bmatrix} = \begin{bmatrix} 1 \\ e^{jkd \cos \theta} \\ \vdots \\ e^{jk(N-1)d \cos \theta} \end{bmatrix}.\tag{27}$$

Thus, the signal correlation matrix can be written as

$$\begin{aligned}\mathbf{R} &= \mathbf{E}[\mathbf{B}^* \mathbf{A}^* \mathbf{A}^t \mathbf{B}^t] + \mathbf{E}[\mathbf{W}^* \mathbf{W}^t] \\ &= \mathbf{B}^* \mathbf{R}_a \mathbf{B}^t + \mathbf{R}_w\end{aligned}\tag{23}$$

where t stands for the complex conjugate transpose. For an equally spaced array with N elements located at $0, d, 2d, \dots, (N-1)d$ the size of the correlation matrix in Eq. 23 is $N \times N$. In this case, the maximum number of emitters that can be detected cannot exceed $(N-1)$.

The matrix \mathbf{R} is Hermitian and always positive definite if thermal element noise is present. Thus \mathbf{R} can be diagonalized using a rotation matrix \mathbf{E} where

$$\mathbf{E}\mathbf{R}\mathbf{E}' = \begin{bmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \vdots & \vdots \\ 0 & \cdots & \lambda_N \end{bmatrix}. \quad (28)$$

The matrix \mathbf{E} is unitary ($\mathbf{E}\mathbf{E}'=\mathbf{I}$) and its columns are the eigenvectors of \mathbf{R} , where

$$\mathbf{E} = [\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_N]. \quad (29)$$

The signal covariance matrix is diagonalized by the same rotation matrix. For example, if there is only one signal present ($N=1$), the weight vector for the single signal case is:

$$\mathbf{G} = (\text{constant})\mathbf{e}_1.$$

A measure of the array response is given by the spectral estimator

$$P(\theta) = \frac{1}{|\mathbf{G}'\mathbf{U}(\theta)|^2} = \frac{1}{|\mathbf{e}_1\mathbf{U}(\theta)|^2}. \quad (30)$$

A large value of P denotes high array sensitivity; that is a pattern notch or null. Note that this is not the same condition that provides good signal to noise for a conventional array. If more than one emitter is present the spectral estimator becomes

$$P(\theta) = \frac{1}{\left| [\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_{N-M}]' \mathbf{U}(\theta) \right|^2} \quad (31)$$

The power response for the conventional equally spaced array of 6 elements using ordinary beamforming is shown in Figure 7. In this case there is no ability to resolve two signals, the null locations are fixed relative to each other. Figure 8 shows the response of a Caratheodory array [Ref. 1] with four elements located at 0, d , $5d/2$, and $3d$. The emitter directions are at 20, 50, 90, 130 and 160 degrees respectively. Note that all of the emitters are essentially resolved, but the sensitivity is only 10dB in some cases.

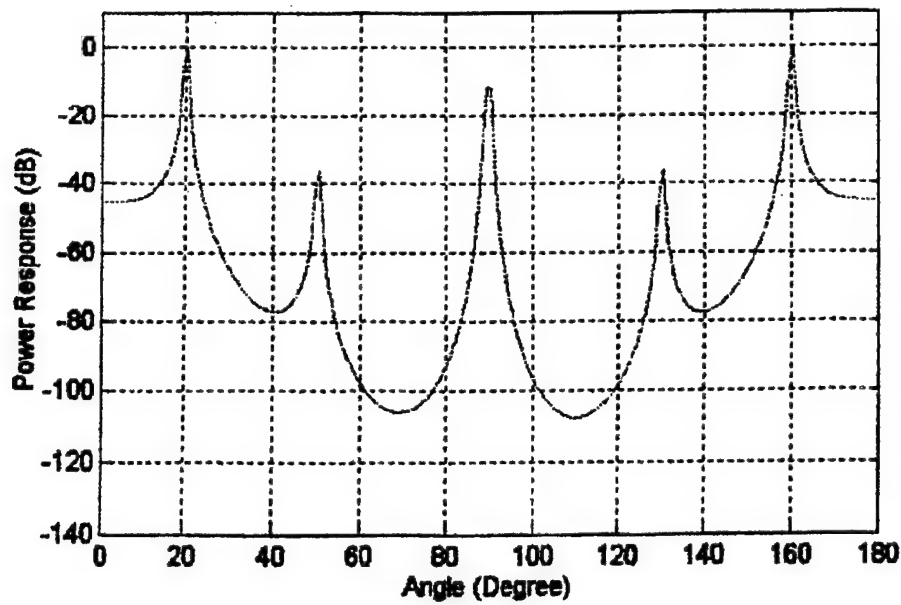


Figure 7. Conventional Equally Spaced Array with Six Elements when the Emitter Directions are 20, 50, 90, 130, and 160

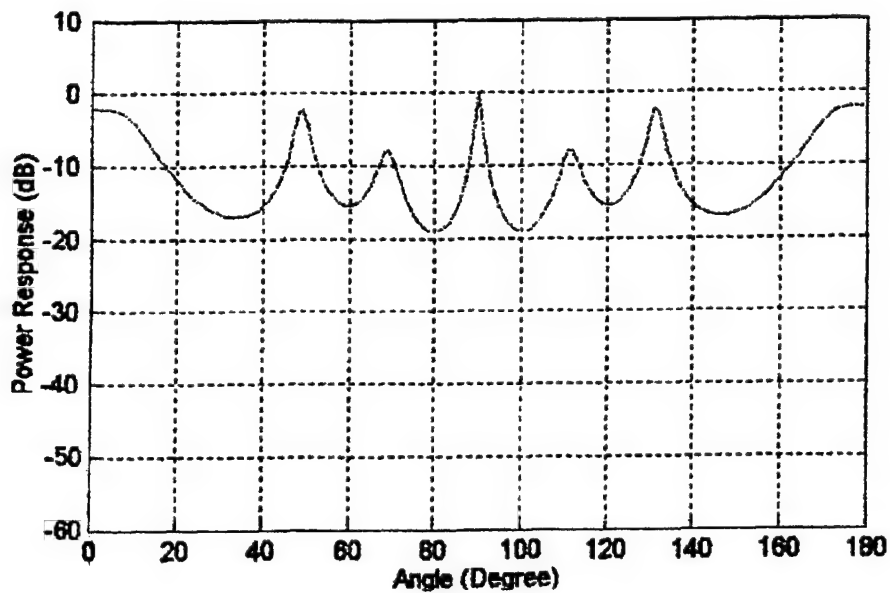


Figure 8. Conventional Processing Scheme for a Caratheodory-array with Four Elements when the Emitter Directions are 20, 50, 90, 130, and 160

IV. RESIDUE ANTENNA ARCHITECTURE

A. OVERVIEW OF RESIDUE NUMBER SYSTEMS

In the first century A.D. the Chinese scholar Sun Tzu authored a book posing the problem of determining a number having remainder 2, 3, and 3 when divided by 3, 4, and 5 respectively. The answer is 23 and is found by using a three-moduli (3,4,5) number system. Later this theorem was called the Chinese Remainder Theorem [Ref. 8]. Eventually this theorem became an important cornerstone in the modern theory of residue number system (RNS) arithmetic.

Digital systems that use residue arithmetic units play an important role in ultra-high speed, dedicated, real-time systems that support pure parallel processing of integer valued data. Residue arithmetic performs addition, subtraction, and multiplication as concurrent operations, side-stepping one of the principal arithmetic delays: managing carry information. Some applications such as the fast Fourier transform, digital filtering, and image processing utilize the efficiencies of RNS arithmetic in addition and multiplication.

The RNS has been employed efficiently in the implementation of digital processors. The RNS is composed of a number of moduli m_i . The integers within each RNS modulus are representative of a sawtooth waveform with the period of the waveform equal to the modulus.

Figure 9 shows the RNS folding waveform and the corresponding RNS states for the moduli $m_1=3$, $m_2=4$, $m_3=5$. In this scheme, given m_i , the integer values within the individual modulus are given by the row vector:

$$[0, 1, \dots, m_i - 1] \quad (32)$$

The required number of comparators for the i th channel is m_i-1 . Notice that all the integers within Eq. 32 form a complete system of length m_i . The dynamic range of this scheme, as well as the number of quantization levels, is given by:

$$\text{Dynamic Range} = \prod_{i=1}^N m_i \quad (33)$$

Table 1 illustrates the RNS encoding values for the moduli (3,4,5).

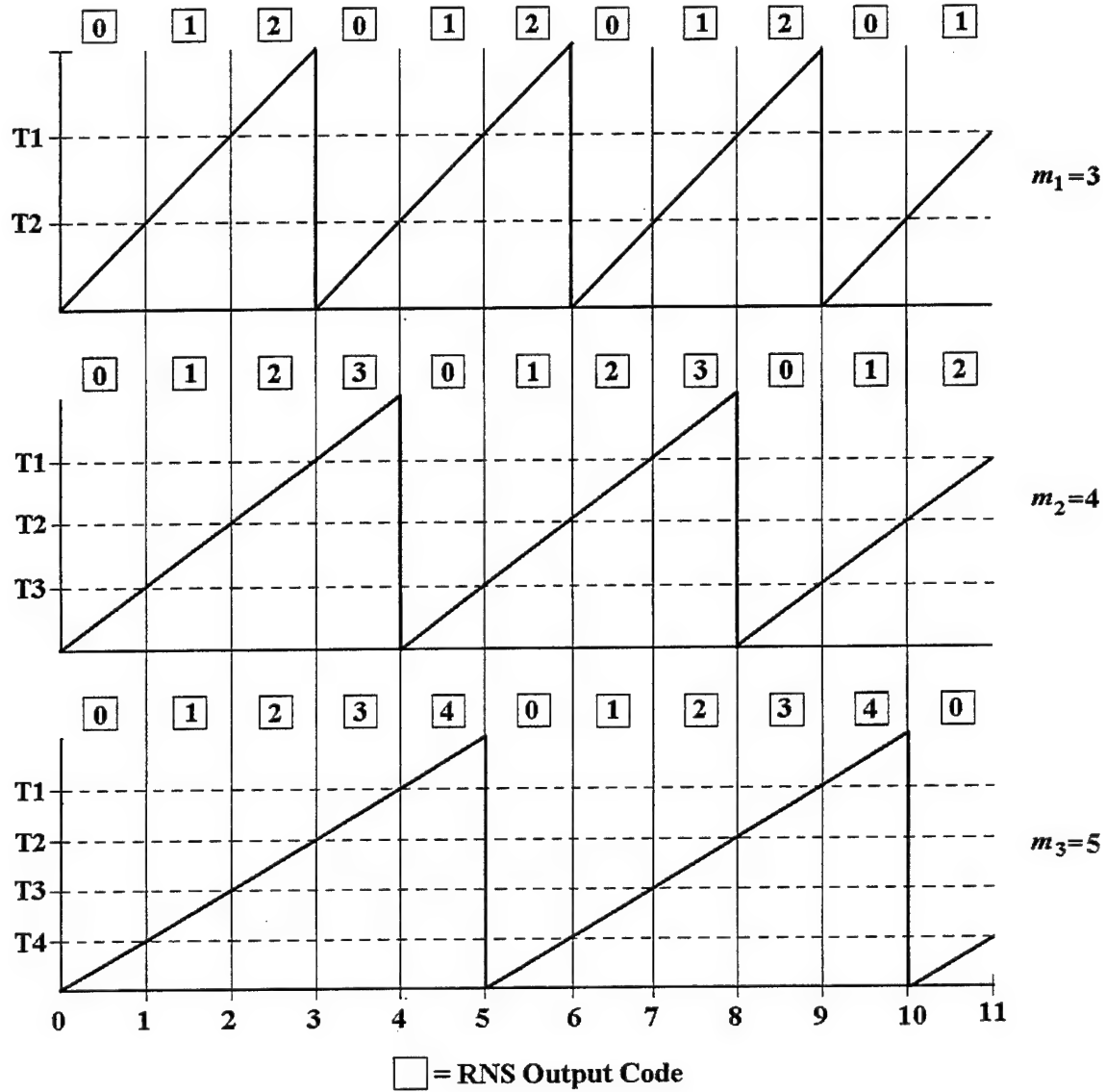


Figure 9. RNS Folding Waveforms and Output Code for the Moduli (3,4,5)

Analog Input	Moduli		
	$m_1=3$	$m_2=4$	$m_3=5$
0	0	0	0
1	1	1	1
2	2	2	2
3	0	3	3
4	1	0	4
5	2	1	0
6	0	2	1
7	1	3	2
8	2	0	3
9	0	1	4
10	1	2	0
11	2	3	1
12	0	0	2
•	•	•	•
•	•	•	•
•	•	•	•
47	2	3	2
48	0	0	3
49	1	1	4
50	2	2	0
51	0	3	1
52	1	0	2
53	2	1	3
54	0	2	4
55	1	3	0
56	2	0	1
57	0	1	2
58	1	2	3
59	2	3	4
60	0	0	0

Table 1. RNS Encoding Procedure for the Moduli (3,4,5)

B. RNS ANTENNA ARCHITECTURE

In the RNS antenna architecture shown in Figure 10, phase detectors are used to combine the individual pairs of elements. The comparator threshold levels and subsequent

logic block are particular to the RNS. The phase response ($\Delta\phi$) of each interferometer corresponds to a sawtooth folding waveform.

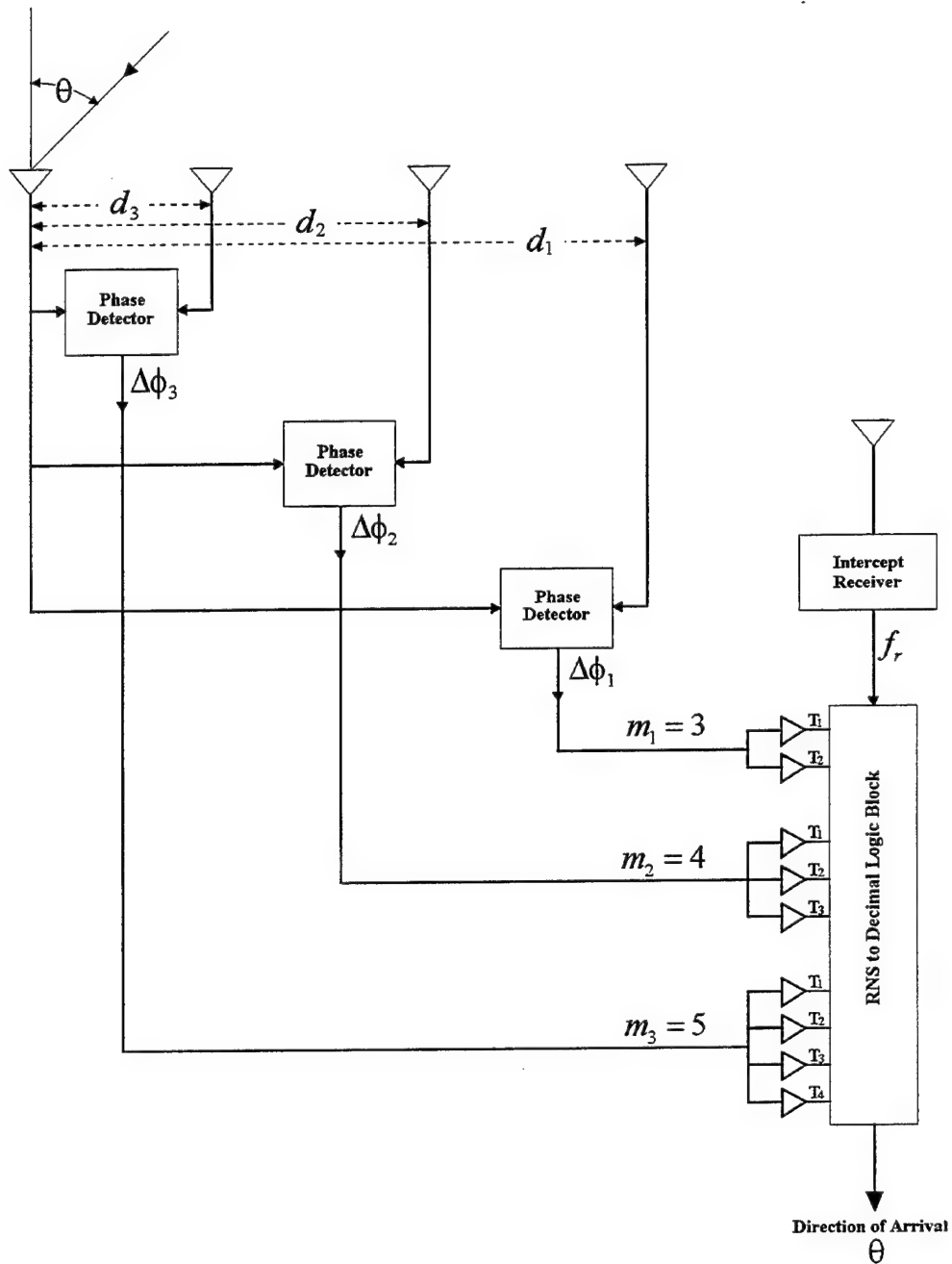


Figure 10. RNS Antenna Architecture

For a given modulus m_i , the number of folds needed to apply the RNS code within the dynamic range is given by:

$$n_{i,\text{RNS}} = \frac{\text{dynamic range}}{m_i} = \frac{M}{m_i} . \quad (34)$$

From Figure 10

$$\begin{aligned} \theta = 0^\circ &\Rightarrow \Delta\phi = 0^\circ , \\ \theta = 90^\circ &\Rightarrow \Delta\phi = \frac{2\pi d}{\lambda} , \\ \theta = -90^\circ &\Rightarrow \Delta\phi = -\frac{2\pi d}{\lambda} . \end{aligned} \quad (35)$$

For a DOA of 180° , the phase response corresponds to a change in phase of 2π . We require that

$$\Delta\phi(90^\circ) - \Delta\phi(-90^\circ) = 2\pi , \quad (36)$$

which implies

$$\frac{4\pi d}{\lambda} = 2\pi, \quad \text{or} \quad d = \frac{\lambda}{2} . \quad (37)$$

Finally, the distance between elements is given by:

$$d_{i,\text{RNS}} = n_{i,\text{RNS}} \left(\frac{\lambda}{2} \right) = \frac{M}{m_i} \left(\frac{\lambda}{2} \right) . \quad (38)$$

Using the same moduli set with center frequency of $f_0 = 8.5$ GHz, the distances for each pair of elements are:

$$\begin{aligned} d_1 &= 10\lambda , \\ d_2 &= 7.5\lambda , \\ d_3 &= 6.0\lambda . \end{aligned}$$

These interferometer distances are used to investigate numerically the performance of the RNS antenna architecture.

C. RNS ARRAY CORRELATION MATRIX FOR MULTIPLE EMITTERS

In this section we briefly investigate the capability of the RNS array with $m_1=3$, $m_2=4$, and $m_3=5$ to resolve multiple emitters at the same frequency. The same approach is used for as the minimum redundancy array (Chapter 2, Section C). For a minimum redundancy array, not all elements must be present, but all correlation lags are present (d , $2d$, $3d$, ..., $(N-1)d$) since all spacings are present at least once. Therefore the array correlation matrix can be filled and consequently up to $\frac{N(N-1)}{2}$ emitters can be resolved with N elements.

For the RNS array not all elements spacings are present. In the case of $m_1=3$, $m_2=4$ and $m_3=5$ the spacings that are present are $0d$, $12d$ (6λ), $15d$ (7.5λ), $20d$ (10λ), $3d$ (1.5λ), $5d$ (2.5λ), and $8d$ (4λ) as shown in Figure 11.

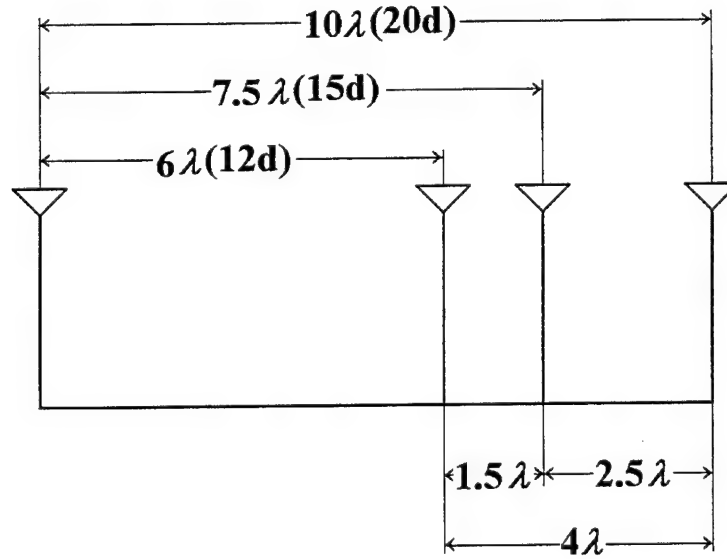


Figure 11. RNS Antenna Spacing for the (3,4,5) Array

The standard periodic array would have a completely filled 21×21 correlation matrix. Not all spacings are represented in the RNS array, and if the missing correlation lags are set to zero (or noise) the matrix would be:

$$R = \begin{bmatrix} R_0 & 0 & 0 & R_3 & 0 & R_5 & 0 & 0 & R_8 & 0 & 0 & 0 & R_{12} & 0 & 0 & R_{15} & 0 & 0 & 0 & 0 & R_{20} \\ 0 & R_0 & 0 & 0 & R_3 & 0 & R_5 & 0 & 0 & R_8 & 0 & 0 & 0 & R_{12} & 0 & 0 & R_{15} & 0 & 0 & 0 & 0 \\ 0 & 0 & R_0 & 0 & 0 & R_3 & 0 & R_5 & 0 & 0 & R_8 & 0 & 0 & 0 & R_{12} & 0 & 0 & R_{15} & 0 & 0 & 0 \\ R_3^* & 0 & 0 & R_0 & 0 & 0 & R_3 & 0 & R_5 & 0 & 0 & R_8 & 0 & 0 & 0 & R_{12} & 0 & 0 & R_{15} & 0 & 0 \\ 0 & R_3^* & 0 & 0 & R_0 & 0 & 0 & R_3 & 0 & R_5 & 0 & 0 & R_8 & 0 & 0 & 0 & R_{12} & 0 & 0 & R_{15} & 0 \\ R_5^* & 0 & R_3^* & 0 & 0 & R_0 & 0 & 0 & R_3 & 0 & R_5 & 0 & 0 & R_8 & 0 & 0 & 0 & R_{12} & 0 & 0 & R_{15} \\ 0 & R_5^* & 0 & R_3^* & 0 & 0 & R_0 & 0 & 0 & R_3 & 0 & R_5 & 0 & 0 & R_8 & 0 & 0 & 0 & R_{12} & 0 & 0 \\ 0 & 0 & R_5^* & 0 & R_3^* & 0 & 0 & R_0 & 0 & 0 & R_3 & 0 & R_5 & 0 & 0 & R_8 & 0 & 0 & 0 & R_{12} & 0 \\ R_8^* & 0 & 0 & R_5^* & 0 & R_3^* & 0 & 0 & R_0 & 0 & 0 & R_3 & 0 & R_5 & 0 & 0 & R_8 & 0 & 0 & 0 & R_{12} \\ 0 & R_8^* & 0 & 0 & R_5^* & 0 & R_3^* & 0 & 0 & R_0 & 0 & 0 & R_3 & 0 & R_5 & 0 & 0 & R_8 & 0 & 0 & 0 \\ 0 & 0 & R_8^* & 0 & 0 & R_5^* & 0 & R_3^* & 0 & 0 & R_0 & 0 & 0 & R_3 & 0 & R_5 & 0 & 0 & R_8 & 0 & 0 \\ 0 & 0 & 0 & R_8^* & 0 & 0 & R_5^* & 0 & R_3^* & 0 & 0 & R_0 & 0 & 0 & R_3 & 0 & R_5 & 0 & 0 & R_8 & 0 \\ R_{12}^* & 0 & 0 & 0 & R_8^* & 0 & 0 & R_5^* & 0 & R_3^* & 0 & 0 & R_0 & 0 & 0 & R_3 & 0 & R_5 & 0 & 0 & R_8 \\ 0 & R_{12}^* & 0 & 0 & 0 & R_8^* & 0 & 0 & R_5^* & 0 & R_3^* & 0 & 0 & R_0 & 0 & 0 & R_3 & 0 & R_5 & 0 & 0 \\ 0 & 0 & R_{12}^* & 0 & 0 & 0 & R_8^* & 0 & 0 & R_5^* & 0 & R_3^* & 0 & 0 & R_0 & 0 & 0 & R_3 & 0 & R_5 & 0 \\ R_{15}^* & 0 & 0 & R_{12}^* & 0 & 0 & 0 & R_8^* & 0 & 0 & R_5^* & 0 & R_3^* & 0 & 0 & R_0 & 0 & 0 & R_3 & 0 & R_5 \\ 0 & R_{15}^* & 0 & 0 & R_{12}^* & 0 & 0 & 0 & R_8^* & 0 & 0 & R_5^* & 0 & R_3^* & 0 & 0 & R_0 & 0 & 0 & R_3 & 0 \\ 0 & 0 & R_{15}^* & 0 & 0 & R_{12}^* & 0 & 0 & 0 & R_8^* & 0 & 0 & R_5^* & 0 & R_3^* & 0 & 0 & R_0 & 0 & 0 & R_3 \\ 0 & 0 & 0 & R_{15}^* & 0 & 0 & R_{12}^* & 0 & 0 & 0 & R_8^* & 0 & 0 & R_5^* & 0 & R_3^* & 0 & 0 & R_0 & 0 & 0 \\ 0 & 0 & 0 & 0 & R_{15}^* & 0 & 0 & R_{12}^* & 0 & 0 & 0 & R_8^* & 0 & 0 & R_5^* & 0 & R_3^* & 0 & 0 & R_0 & 0 \\ R_{20}^* & 0 & 0 & 0 & 0 & R_{15}^* & 0 & 0 & R_{12}^* & 0 & 0 & 0 & R_8^* & 0 & 0 & R_5^* & 0 & R_3^* & 0 & 0 & R_0 \end{bmatrix}$$

Figure 12 shows the array response for 50, 90 and 130 degrees. As expected the performance is not good because of the large number of zeros in the array.

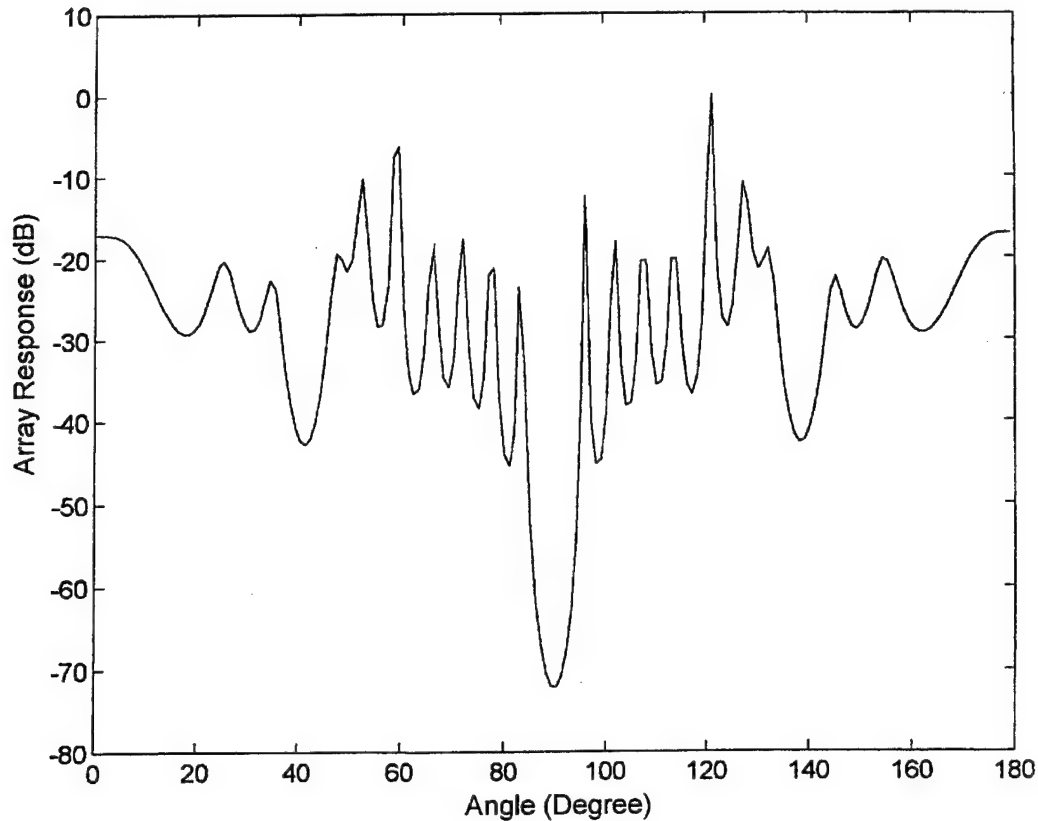


Figure 12. RNS Array Response for Multiple Emitters when the Emitter Directions are 50, 90, and 130

D. COMPUTER SIMULATION OF RNS CODE

The quantization of the phase response from each pair of elements is made separately using a small bank of comparators. The threshold levels for each comparator are shown in Table 2. If the phase response is plotted as a function of $\sin\theta$, an ideal sawtooth waveform is obtained as shown at the top of Figure 13. An actual plot of the phase response is shown in Figure 14 as a function of the input direction of arrival θ . The sawtooth waveform stretches out as θ , the input direction of arrival, goes from 0 to 90°. For a specific input direction of arrival, there will be a corresponding integer number of comparators in the "on" state within the modulus m_i . The value of the integers depends on the measured phase difference associated with each pair of elements for the specific input direction of arrival.

The three RNS codes (one for each pair of antenna elements) are indices to an RNS logic look-up table that gives the resolved direction of arrival. Table 3 shows the three RNS codes and the resolved direction of arrival. The numbered codes represent the number of comparators in the "on" state.

Moduli	T_1	T_2	T_3	T_4
3	-1.047	1.047		
4	-1.571	0.00	1.571	
5	-1.885	-0.628	0.628	1.885

Table 2. Threshold Levels for the Moduli (3,4,5)

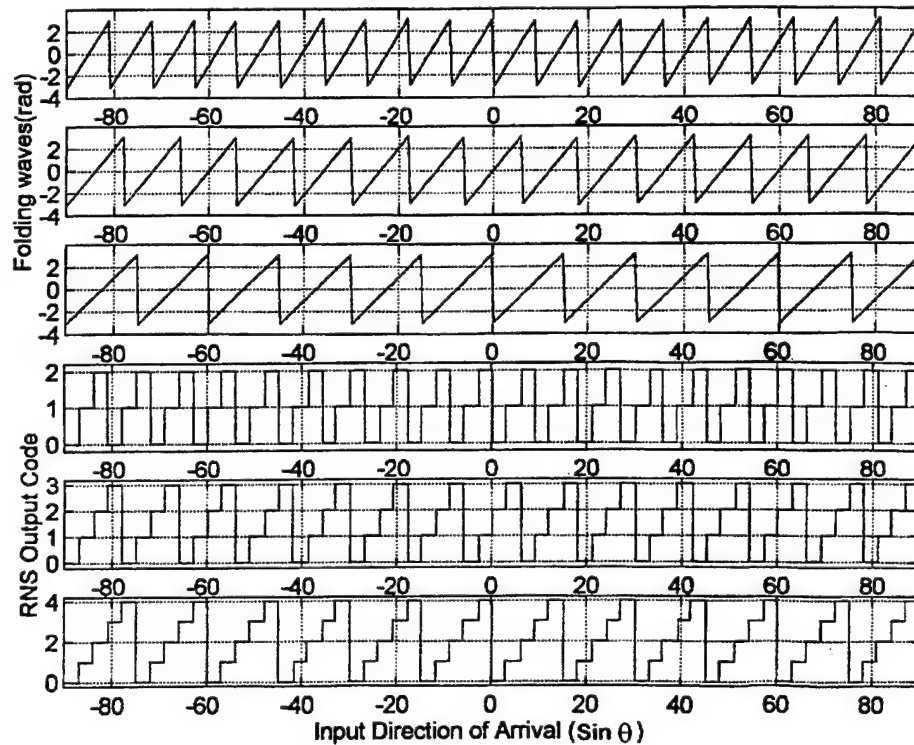


Figure 13. Phase Response as a Function of $\sin\theta$ for the Moduli (3,4,5)

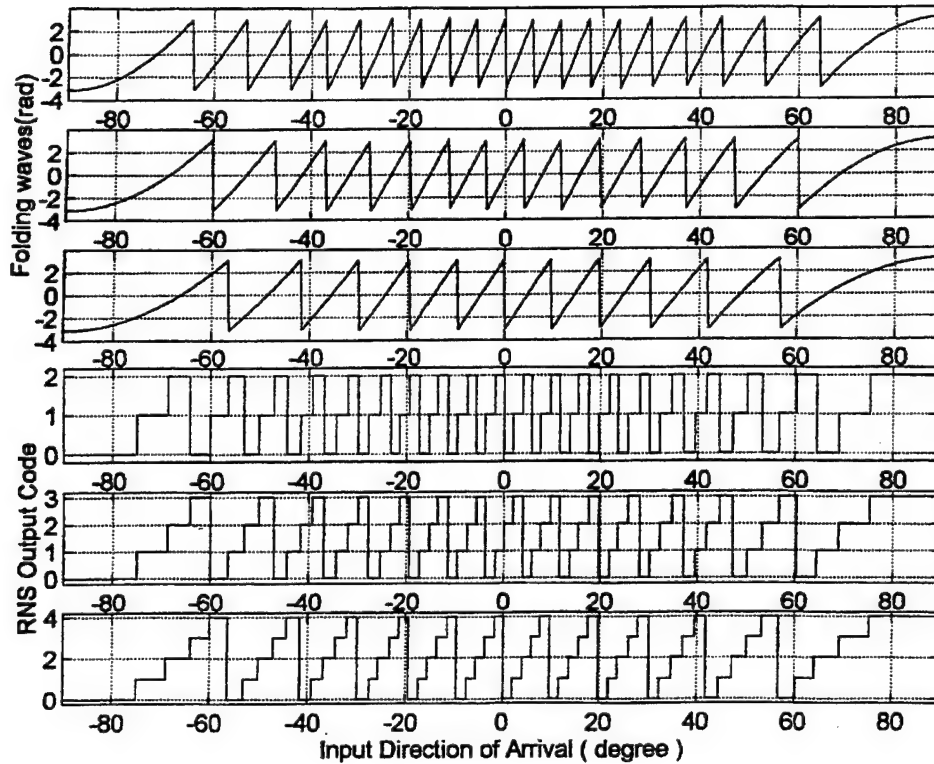


Figure 14. Phase Response as a Function of the Input Direction of Arrival (θ) for the Moduli (3,4,5)

Figure 15, shows a plot of the resolved direction of arrival as a function of θ , the input direction of arrival. The RNS antenna system gives the direction of arrival information over the complete field of view without ambiguity (DOA increment is 0.3°). However the accuracy varies depending on the actual direction of arrival of the incoming signal. Notice also that the quantization width increases as the direction of arrival approaches $\pm 90^\circ$.

3 RNS Codes	Resolved DOA (degrees)	3 RNS Codes	Resolved DOA (degrees)
0 0 0	-82.5824	0 2 0	0.9551
1 1 1	-72.0627	1 3 1	2.8664
2 2 2	-66.5593	2 0 2	4.7809
0 3 3	-62.1158	0 1 3	6.7007
1 0 4	-58.2581	1 2 4	8.6382
2 1 0	-54.7864	2 3 0	10.5655
0 2 1	-51.5928	0 0 1	12.5152
1 3 2	-48.6111	1 1 2	14.4797
2 0 3	-45.7968	2 2 3	16.4618
0 1 4	-43.1187	0 3 4	18.4644
1 2 0	-40.5534	1 0 0	20.4907
2 3 1	-38.0832	2 1 1	22.5442
0 0 2	-35.6940	0 2 2	24.6287
1 1 3	-33.3745	1 3 3	26.7487
2 2 4	-31.1155	2 0 4	28.9091
0 3 0	-28.9091	0 1 0	31.1155
1 0 1	-26.7487	1 2 1	33.3745
2 1 2	-24.6287	2 3 2	35.6940
0 2 3	-22.5442	0 0 3	38.0832
1 3 4	-20.4907	1 1 4	40.5534
2 0 0	-18.4644	2 2 0	43.1187
0 1 1	-16.4618	0 3 1	45.7968
1 2 2	-14.4797	1 0 2	48.6111
2 3 3	-12.5152	2 1 3	51.5928
0 0 4	-10.5655	0 2 4	54.7864
1 1 0	-8.6382	1 3 0	58.2581
2 2 1	-6.7007	2 0 1	62.1158
0 3 2	-4.7809	0 1 2	66.5593
1 0 3	-2.8664	1 2 3	72.0627
2 1 4	-0.9551	2 3 4	82.5824

Table 3. RNS Code and Equivalent Resolved Direction of Arrival for Modulus 3, 4, and 5

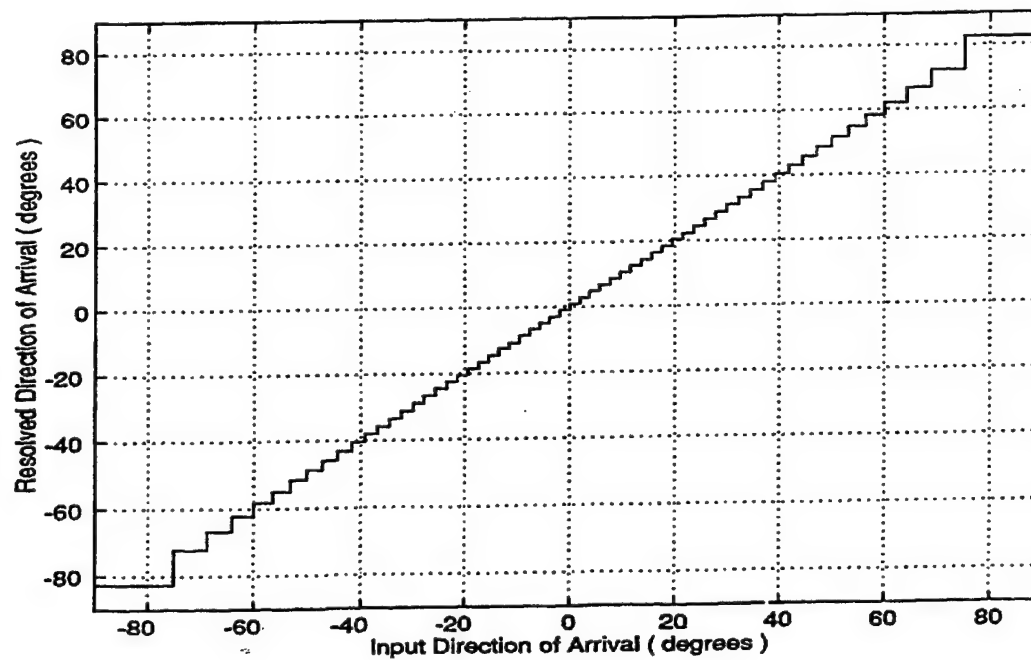


Figure 15. Resolved Direction of Arrival in Function of the Input Direction of Arrival when DOA Increment is 0.3°

V. ENCODING ERROR CORRECTION

A. ENCODING ERRORS

The DOA estimates previously shown in Figure 15 are based on ideal hardware performance. One significant assumption is that all of the comparator thresholds are ideal and cross the falling phase response waveforms together at each DOA code transition point. If the samples are generated by stepping the emitter direction between -90° and 90° in increments of 0.03° , for example, some samples will fall about the code transition point and result in encoding errors due to some comparators do not switch simultaneously. When the emitter is swept through an angular sector, the DOA estimates from a realizable system could contain large spikes or "glitches" at these points. An illustration of the encoding errors is given in Table 4. The columns in Table 4 correspond to the first 5 RNS codes shown in Table 3 with each code sampled three times. The seventh sample shows the nature of an encoding error where some of the comparators in modulus 5 do not switch at the same time. The seventh sample should be $m_1=2$, $m_2=2$ and $m_3=2$. This results in a glitch at this sample within the transfer function as shown in Figure 16.

Sample	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Modulus 3	0	0	0	1	1	1	2	2	2	0	0	0	0	1
Modulus 4	0	0	0	1	1	1	2	2	2	3	3	3	0	0
Modulus 5	0	0	0	1	1	1	1	2	2	3	3	3	4	4

Table 4. Encoding Error in the RNS Output Code

To examine the nature of the encoding errors, a simulation was performed with the threshold levels shifted from the ideal (error-free) threshold levels in Table 2 to a set of new threshold levels. The errors are induced in order to compare three interpolation methods designed to eliminate the encoding errors. The new threshold values are shown in Table 5, and the glitches in the DOA are shown in Figure 17, with the total number of glitches equal to 44.

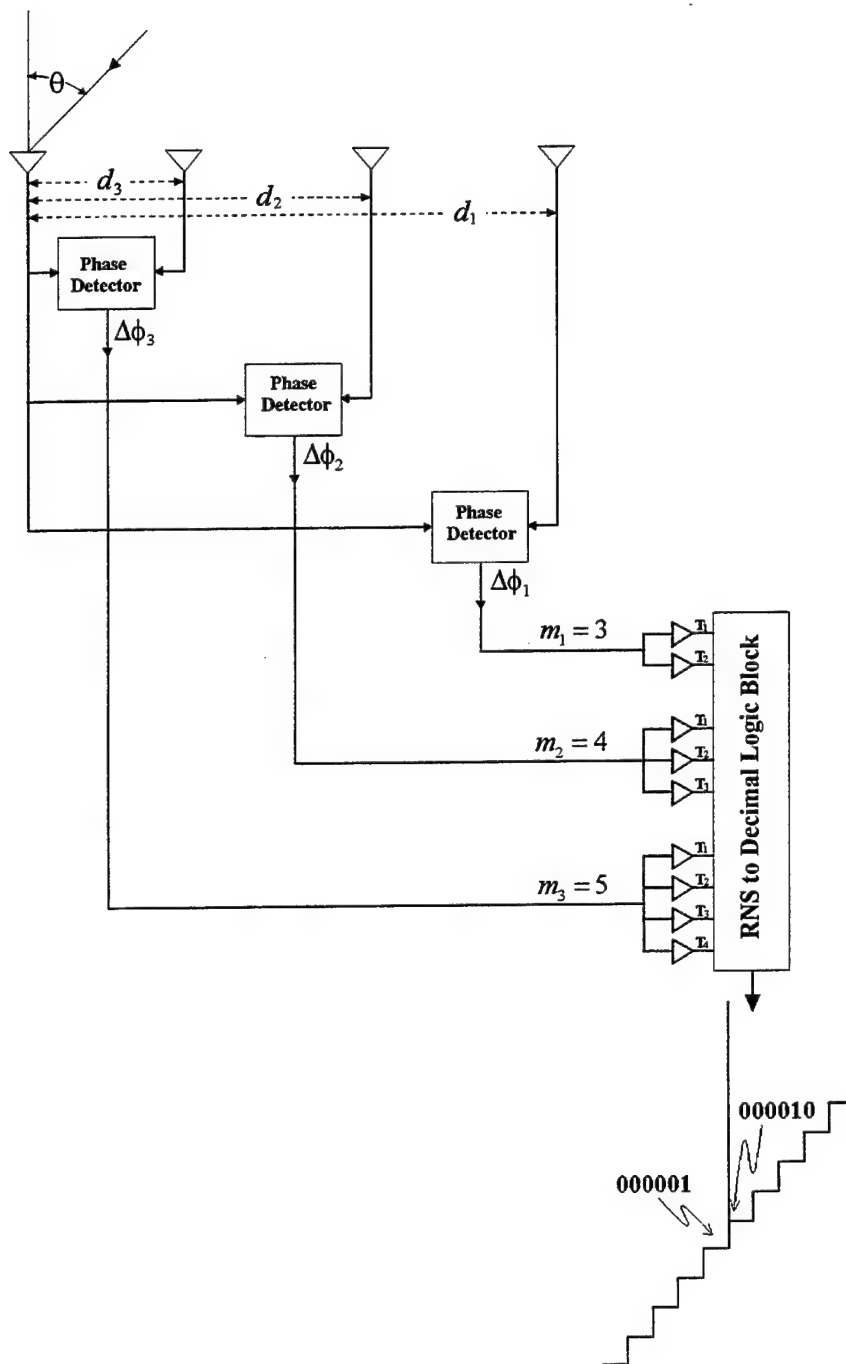


Figure 16. RNS-to-Decimal Logic Block and Glitch

Moduli	T_1	T_2	T_3	T_4
3	-1.047	1.047		
4	-1.560	0.000	1.560	
5	-1.870	-0.640	0.640	1.870

Table 5. New threshold Levels for the Moduli (3,4,5)

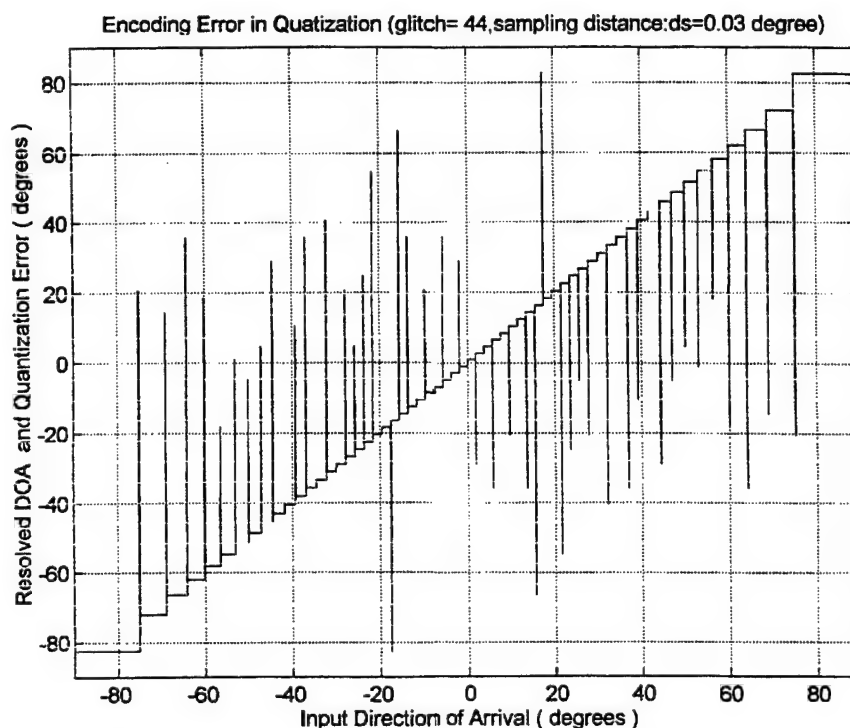


Figure 17. Glitches and Resolved Direction of Arrival

B. ADDITIONAL COMPARATORS TO ISOLATE ENCODING ERRORS

The original threshold levels within the minimum modulus are set to the level of 1.047, -1.047 radians. The phase response for the minimum modulus ($m_i=3$) is shown in Figure 18 and illustrates the position of the comparator threshold levels.

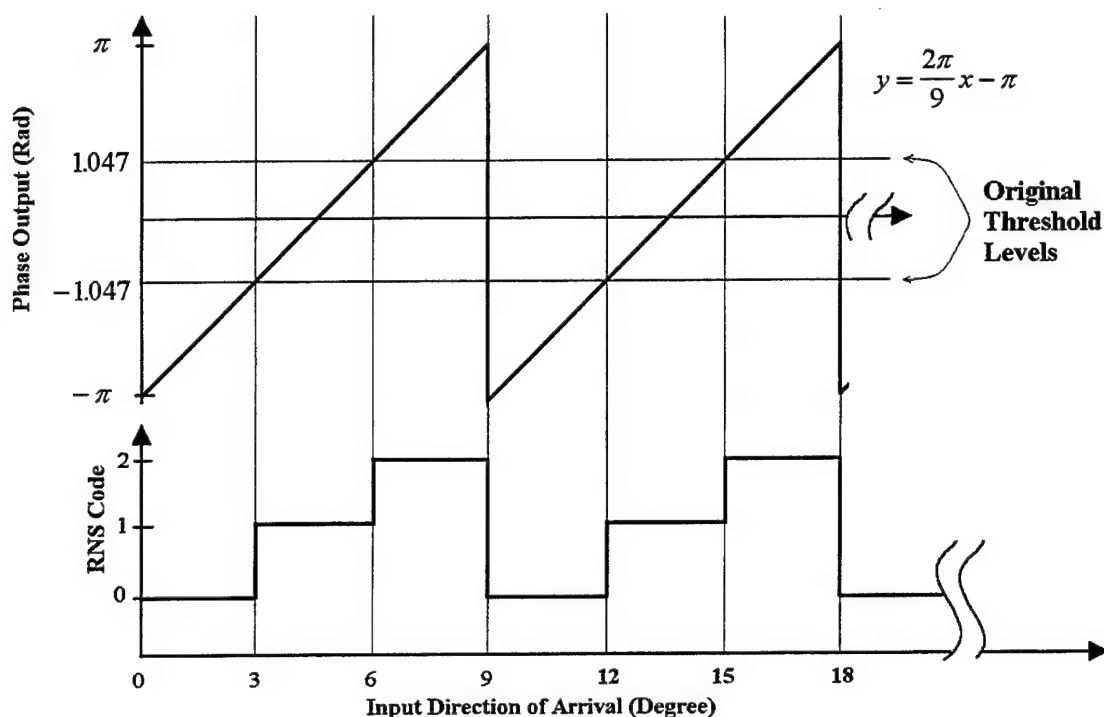


Figure 18. Original Threshold Level in Modulus 3

To isolate the possible encoding errors, $2m_1$ comparators are used in the minimum modulus (instead of $m_1 - 1$). The comparator thresholds levels are such that a small band is set up about the code transition points. The values of the threshold levels depend on the width of the band and can be expressed as a percentage of the least significant angle (LSA) width. The six-threshold levels for modulus $m_1=3$ are shown in Figure 19, and listed in Table 6 as a function of the LSA percentage.

The encoding errors are isolated by comparing the parity value for the $2m_1$ comparators in the minimum modulus. If the parity value is odd number (1, 3 and 5), then the sample falls outside the small transition band and does not present a problem. If the parity is an even number (2, 4 and 6), the sample represents a possible DOA encoding error. The goal of the interpolation method is to determine the best DOA estimate given the fact that the sample could present a possible error.

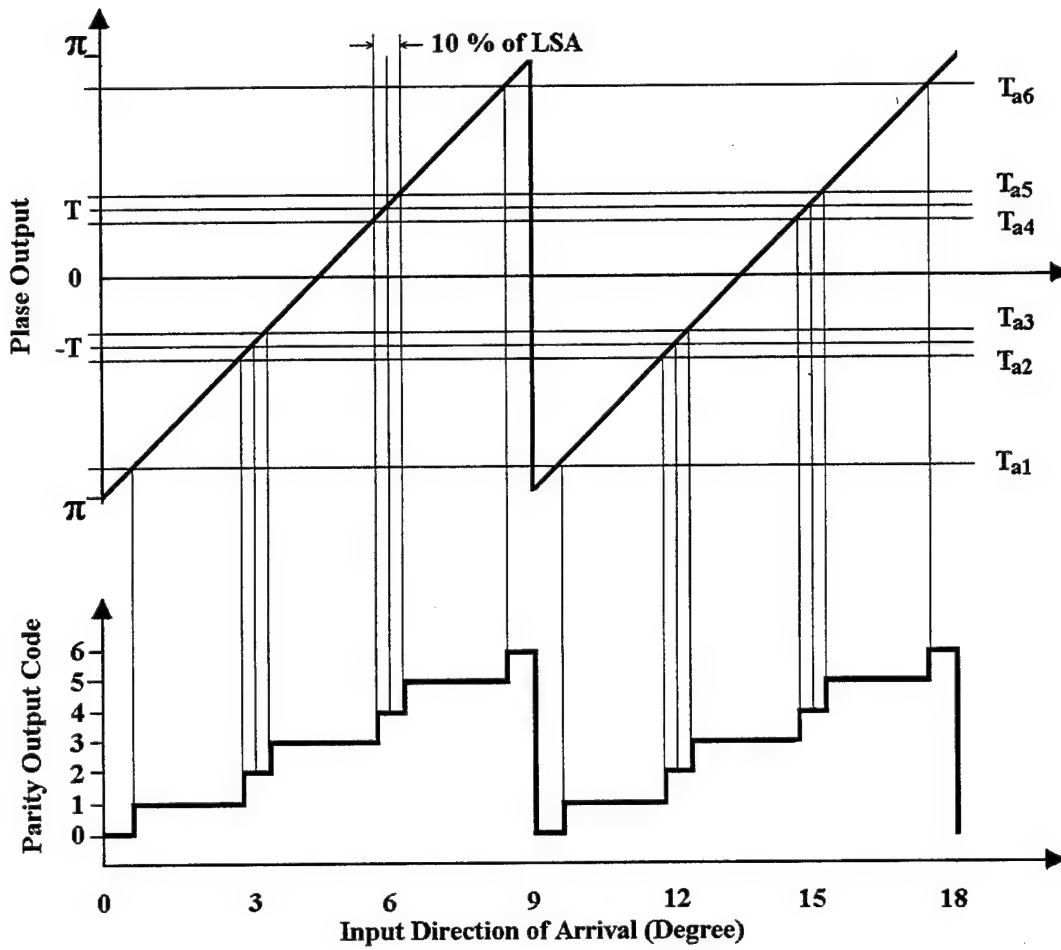


Figure 19. Additional Threshold Levels and RNS Output Code

LSA Percentage	T_{a1}	T_{a2}	T_{a3}	T_{a4}	T_{a5}	T_{a6}
10.00	-3.0369	-1.1519	-0.9425	0.9425	1.1519	3.0369
9.00	-3.0473	-1.1414	-0.9529	0.9529	1.1414	3.0473
8.00	-3.0578	-1.1310	-0.9634	0.9634	1.1310	3.0578
7.00	-3.0683	-1.1205	-0.9739	0.9739	1.1205	3.0683
6.00	-3.0788	-1.1100	-0.9844	0.9844	1.1100	3.0788
5.00	-3.0892	-1.0996	-0.9948	0.9948	1.0996	3.0892
4.00	-3.0997	-1.0891	-1.0053	1.0053	1.0891	3.0997
3.00	-3.1102	-1.0786	-1.0158	1.0158	1.0786	3.1102
2.00	-3.1206	-1.0681	-1.0263	1.0263	1.0681	3.1206
1.00	-3.1311	-1.0577	-1.0367	1.0367	1.0577	3.1311

Table 6. LSA Percentage and Six-Additional Threshold Values

C. INTERPOLATION

A block diagram of the RNS antenna architecture including the interpolation processing is shown in Figure 20.

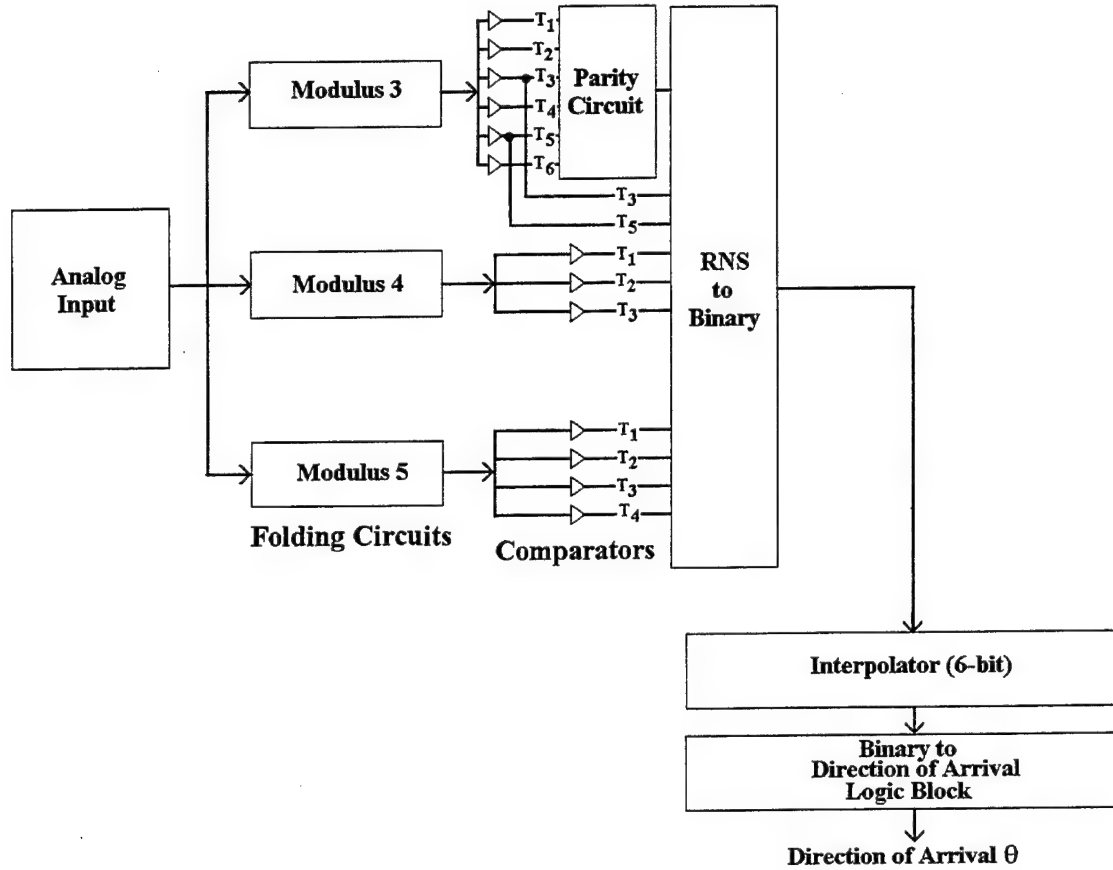


Figure 20. The Procedure of DOA Decoder

The RNS-to-binary logic block is set after the parity circuit as the interpolation procedures are accomplished in the binary number system as described below. The RNS code and corresponding binary values are shown in Table 7. The encoding errors at samples 7 and 13, and the RNS code in Table 4 are converted to a six-bit binary code and are shown in Table 8.

Angle	RNS Code	Six-Bit Binary Code	Angle	RNS Code	Six-Bit Binary Code
-82.5824	0 0 0	000000	0.9551	0 2 0	011110
-72.0627	1 1 1	000001	2.8664	1 3 1	011111
-66.5593	2 2 2	000010	4.7809	2 0 2	100000
-62.1158	0 3 3	000011	6.7007	0 1 3	100001
-58.2581	1 0 4	000100	8.6382	1 2 4	100010
-54.7864	2 1 0	000101	10.5655	2 3 0	100011
-51.5928	0 2 1	000110	12.5152	0 0 1	100100
-48.6111	1 3 2	000111	14.4797	1 1 2	100101
-45.7968	2 0 3	001000	16.4618	2 2 3	100110
-43.1187	0 1 4	001001	18.4644	0 3 4	100111
-40.5534	1 2 0	001010	20.4907	1 0 0	101000
-38.0832	2 3 1	001011	22.5442	2 1 1	101001
-35.6940	0 0 2	001100	24.6287	0 2 2	101010
-33.3745	1 1 3	001101	26.7487	1 3 3	101011
-31.1155	2 2 4	001110	28.9091	2 0 4	101100
-28.9091	0 3 0	001111	31.1155	0 1 0	101101
-26.7487	1 0 1	010000	33.3745	1 2 1	101110
-24.6287	2 1 2	010001	35.6940	2 3 2	101111
-22.5442	0 2 3	010010	38.0832	0 0 3	110000
-20.4907	1 3 4	010011	40.5534	1 1 4	110001
-18.4644	2 0 0	010100	43.1187	2 2 0	110010
-16.4618	0 1 1	010101	45.7968	0 3 1	110011
-14.4797	1 2 2	010110	48.6111	1 0 2	110100
-12.5152	2 3 3	010111	51.5928	2 1 3	110101
-10.5655	0 0 4	011000	54.7864	0 2 4	110110
-8.6382	1 1 0	011001	58.2581	1 3 0	110111
-6.7007	2 2 1	011010	62.1158	2 0 1	111000
-4.7809	0 3 2	011011	66.5593	0 1 2	111001
-2.8664	1 0 3	011100	72.0627	1 2 3	111010
-0.9551	2 1 4	011101	82.5824	2 3 4	111011

Table 7. RNS code and Six-bit Binary Code in Moduli (3,4,5)

Sample	1	2	3	4	5	6	7	8	9	10	11	12	13	14
MSB	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-	0	0	0	0	0	0	0	0	0	0	0	0	1	0
-	0	0	0	0	0	0	0	0	0	0	0	0	1	0
-	0	0	0	0	0	0	1	0	0	0	0	0	0	1
-	0	0	0	0	0	0	1	1	1	1	1	1	0	0
LSB	0	0	0	1	1	1	0	0	0	1	1	1	0	0

MSB represents the Most Significant Bit of the converter

LSB is the Least Significant Bit in the six-bit converter

Table 8. Encoding Error in Six-bit Binary Code

1. LSB Shift Method

In Table 4 glitches occurred at samples 7 and 13. From that binary data the output signal jumps from the value given by 000001 through 000110 to 000010 as shown in Table 8. To resolve these glitches, we investigate the use of a 7×5 Error Correction Block (ECB) matrix. The output of a parity circuit determines whether the sample is good or bad. If the parity output is an odd number (1, 3 and 5) the sample is good, whereas an even number (2, 4 and 6) the output represents a possible bad sample. The parity circuit and error correction block are shown in Figure 21.

a. LSB Shift Algorithm

The parity and the 6-bit word (from the PLA) are clocked in parallel through a 5-state buffer. The buffer allows the 6-bit word to be replaced with an interpolated value depending on the value of the parity. If the parity is odd (good sample), the 6-bit word is clocked through the buffer unaltered. If the parity is even, the values ahead and behind the even parity value are used to interpolate the possible encoding error. The interpolation process compares the most significant bits (MSBs) of the odd parity words stopping at the point where the MSBs do not agree. The MSBs that are the same are copied into the MSBs for the even parity word. The LSBs come directly from the original (even parity) word. Figure 22 shows an example where there is one out of five possible bad samples. In the interpolation process, P_2 and P_0 combine to form the MSBs of T_1 (an intermediate state). The LSBs of T_1 come from P_1 . At time $t+1$, T_1 is shifted into P_2 . The situation where two samples are even parity is shown in Figure 23. In this case P_3 and P_0 combine to form the

MSBs of T_1 and T_2 . The LSBs of T_1 and T_2 are copied directly from P_1 and P_2 . T_1 is shifted into P_2 and T_2 is shifted into P_3 at $t+1$. The interpolation procedure for three out of five samples having even parity is shown in Figure 24. The interpolation procedure for four out of five samples having even parity is shown in Figure 25.

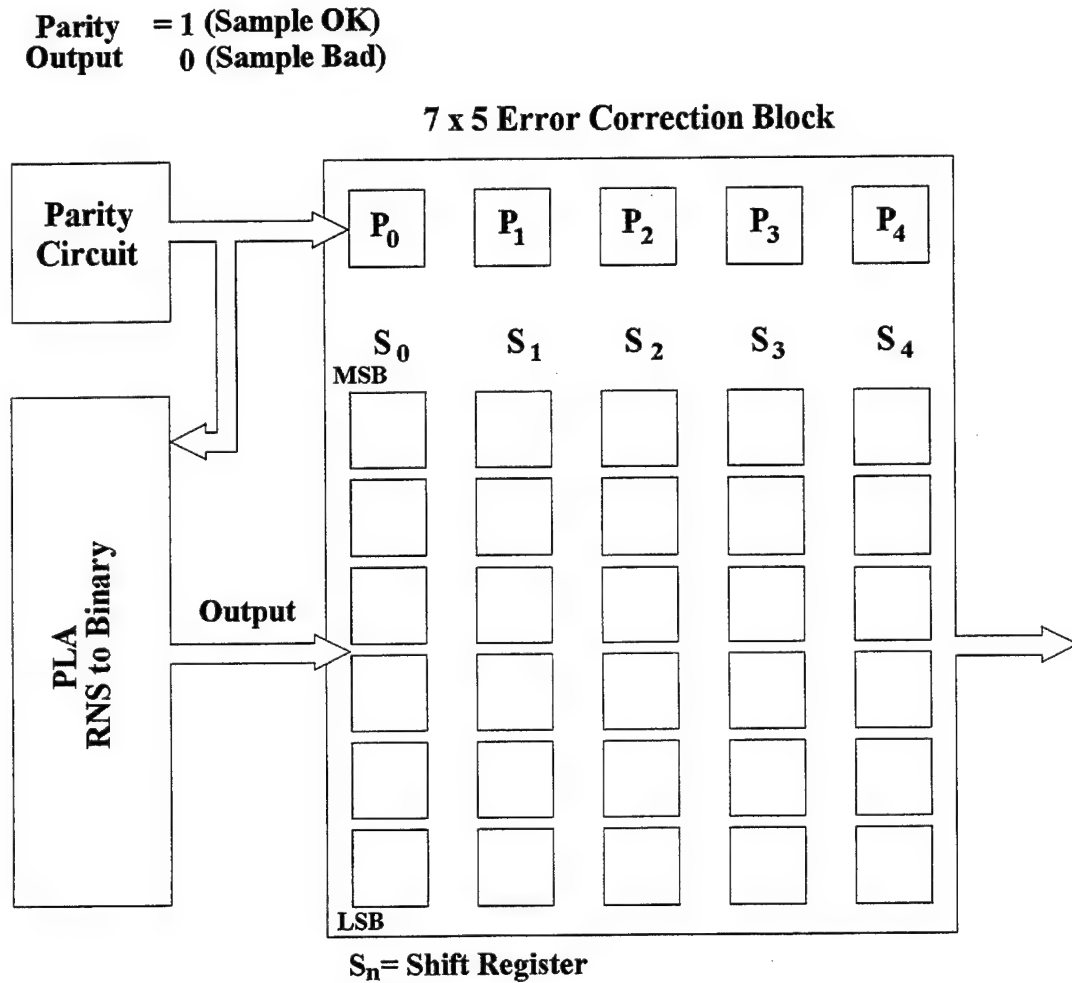


Figure 21. Parity Circuit and Error Correction Block

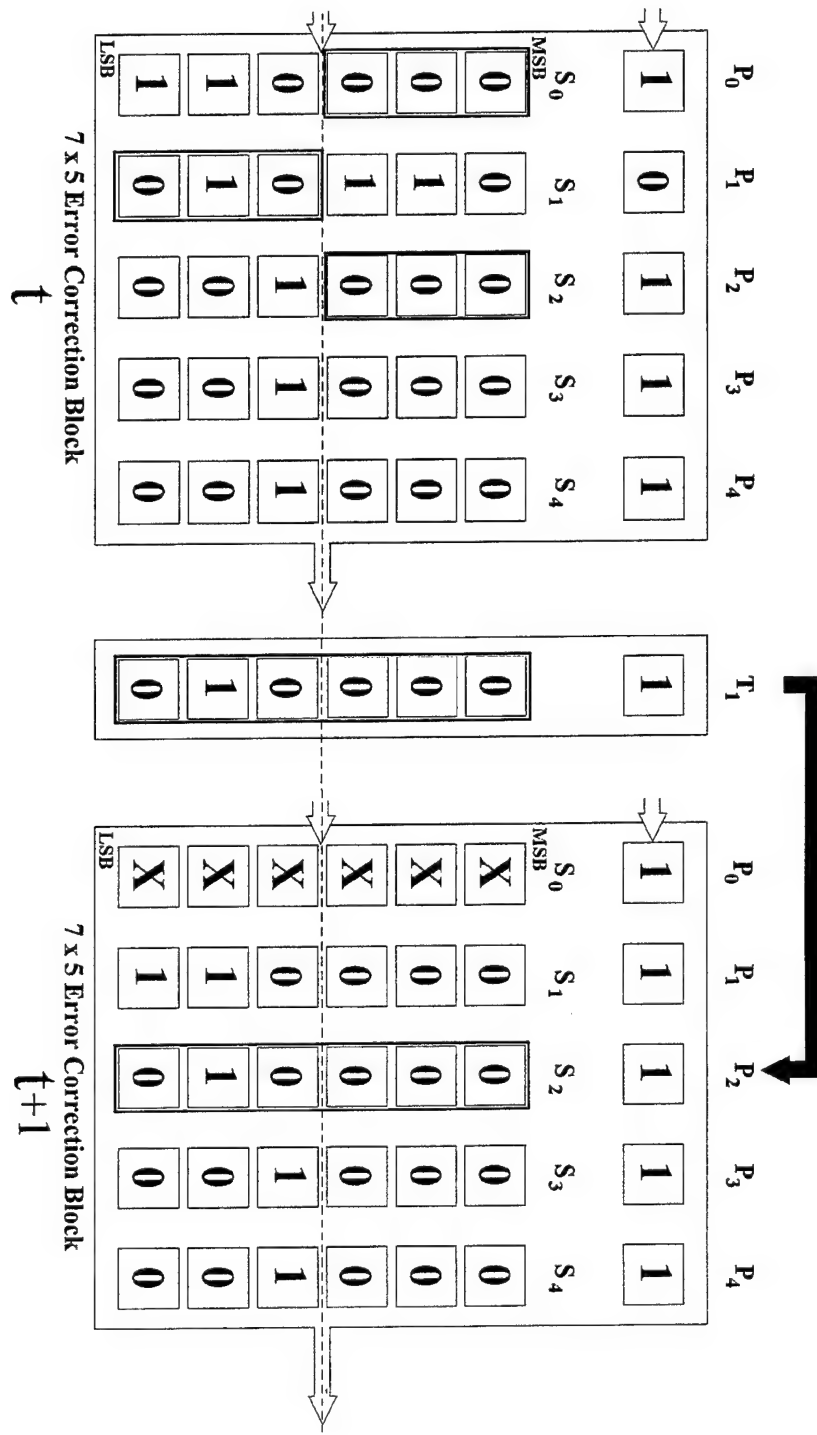


Figure 22. LSB-Shift Method in Case of 1/5 Error Samples

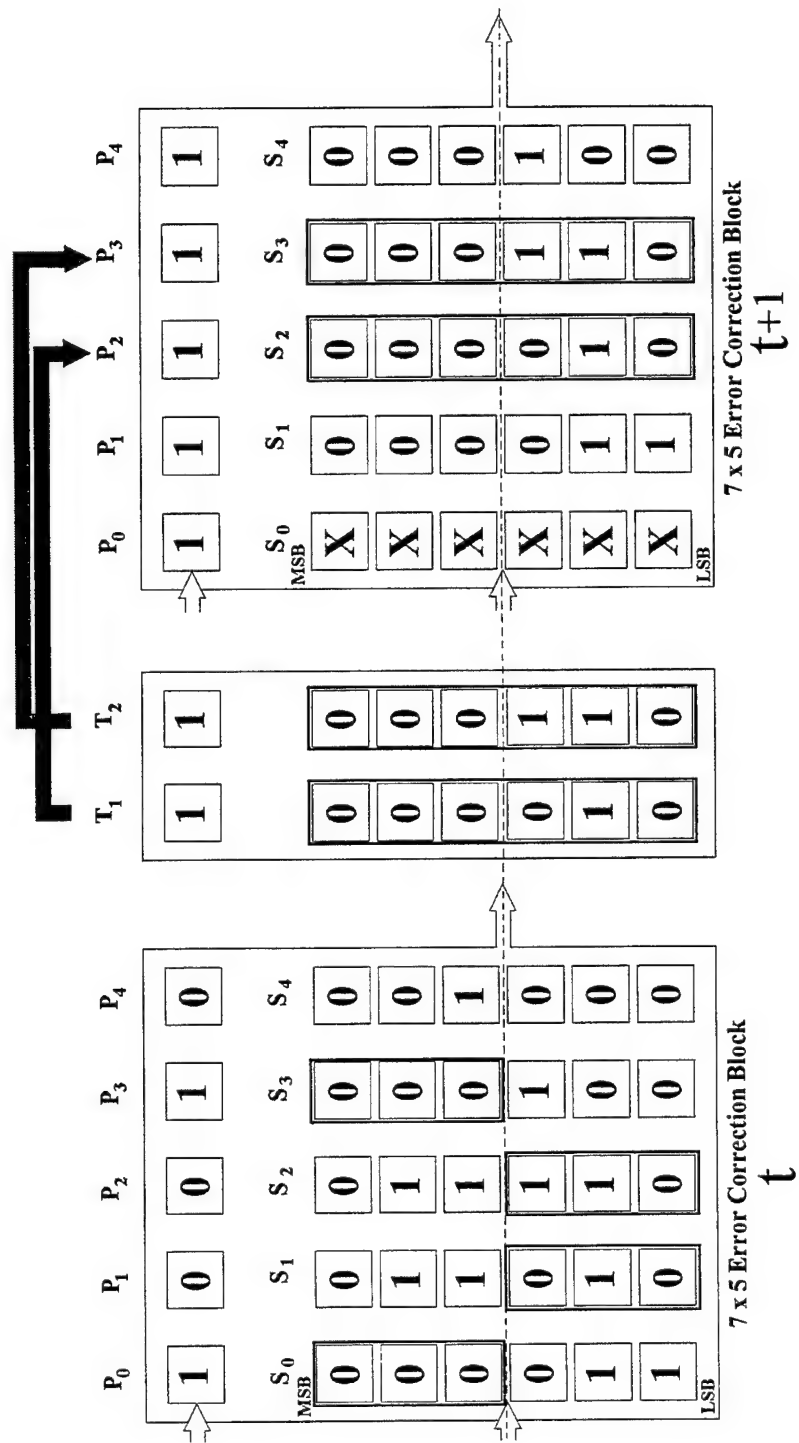


Figure 23. LSB-Shift Method in Case of 2/5 Error Samples

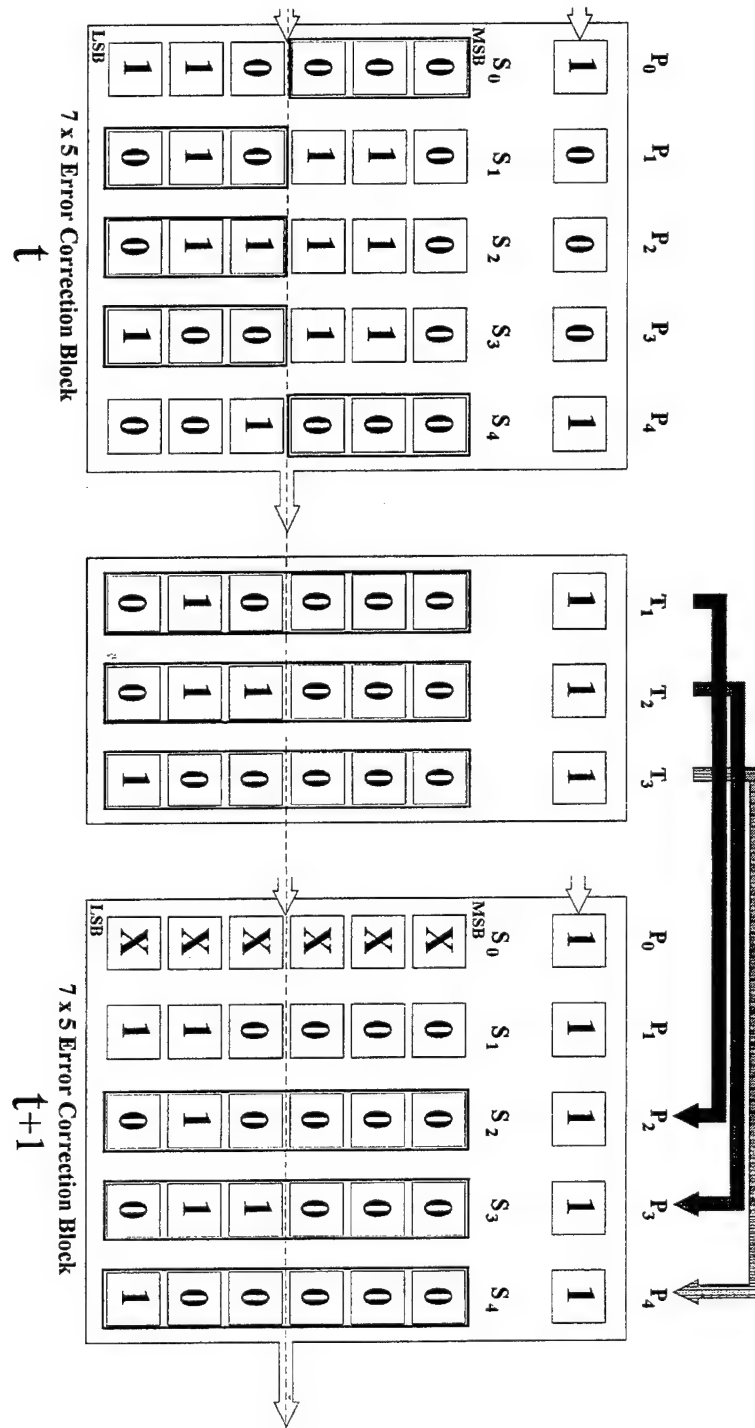


Figure 24. LSB-Shift Method in Case of 3/5 Error Samples

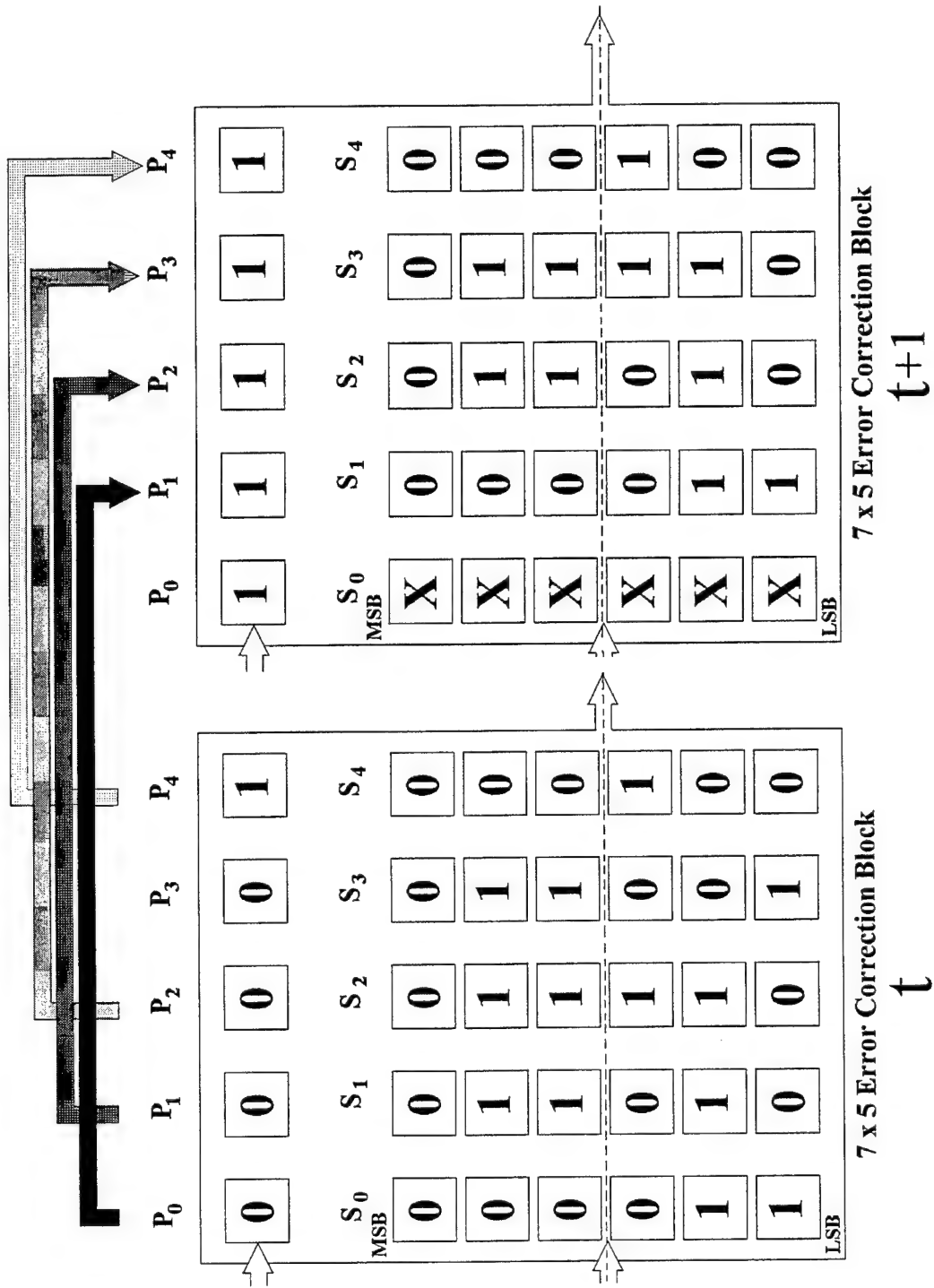


Figure 25. LSB-Shift Method in Case of 4/5 Error Sample Case

b. Simulation Results

The DOA transfer function including the encoding errors was shown in Figure 17. The LSB shift method with the interpolation width equal to 1% of the LSA is shown in Figure 26. The total number of glitches has been reduced from 44 to 42. Figure 27 through Figure 30 show the result of the LSB shift method with the interpolation width ranging from 2% to 5% respectively. Figure 31 summarizes the results and shows the total number of glitches as a function of the isolation bandwidth (percentage of LSA). The larger the bandwidth, the fewer the number of glitches.

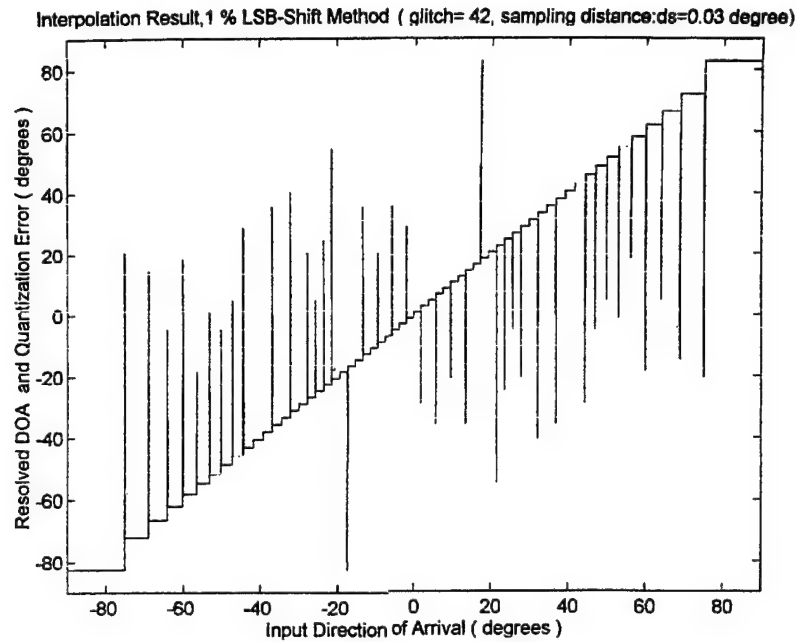


Figure 26. LSB-Shift Interpolation Results, 1% of LSA

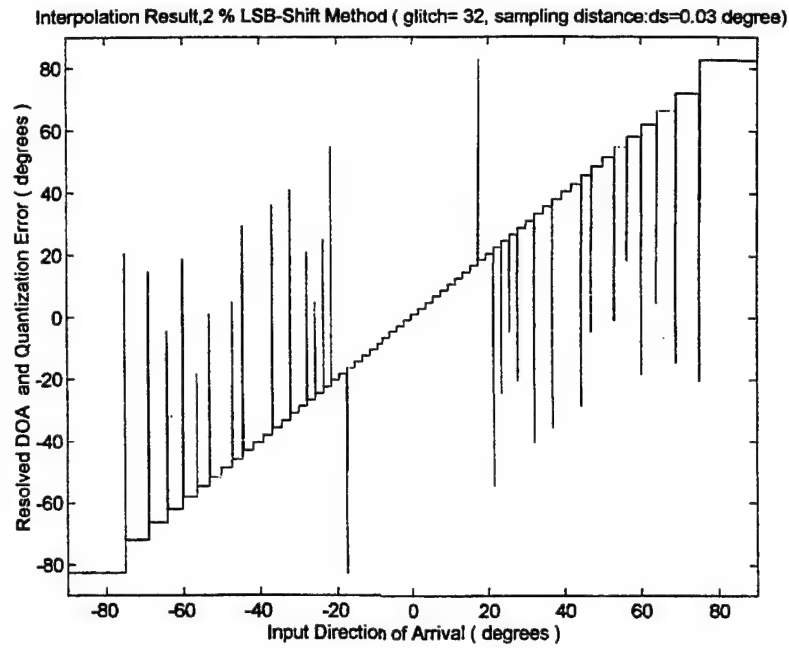


Figure 27. LSB-Shift Interpolation Results, 2% of LSA

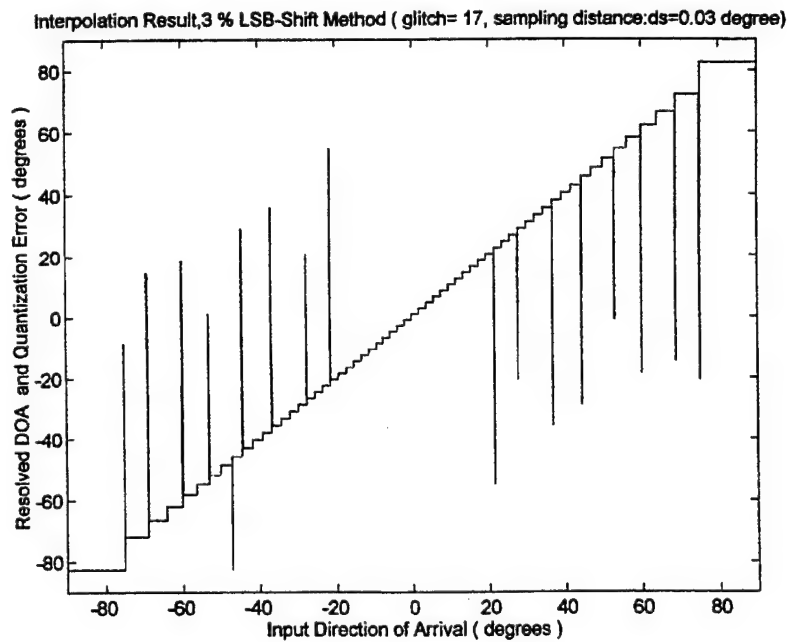


Figure 28. LSB-Shift Interpolation Results, 3% of LSA

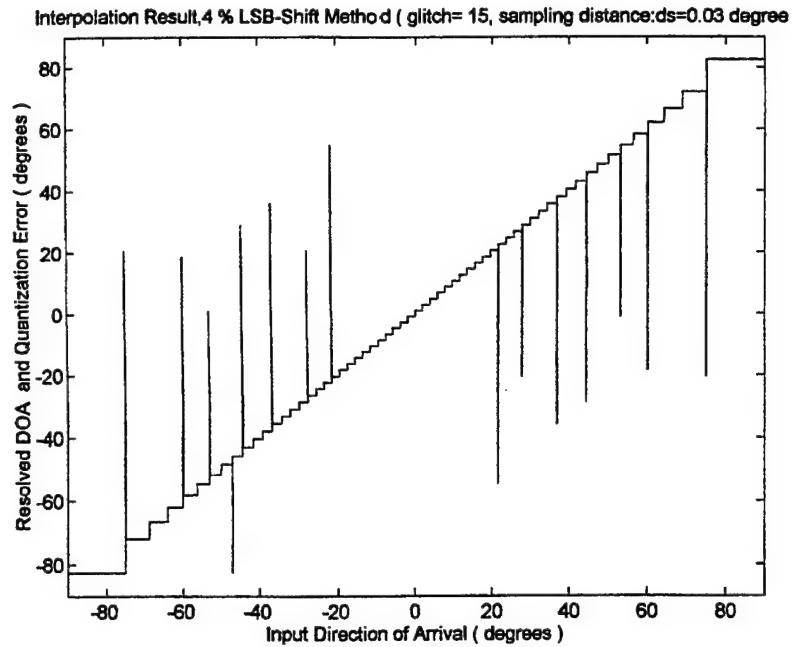


Figure 29. LSB-Shift Interpolation Results, 4% of LSA

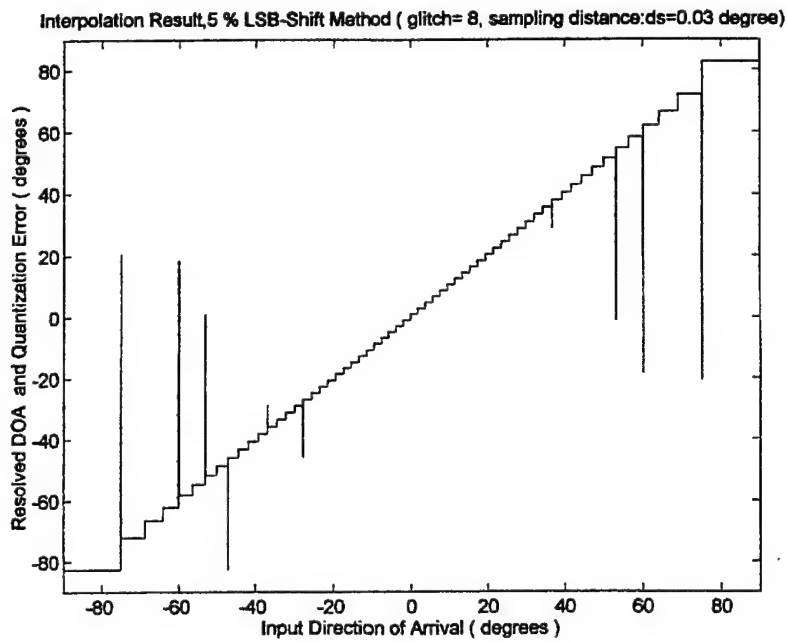


Figure 30. LSB-Shift Interpolation Results, 5% of LSA

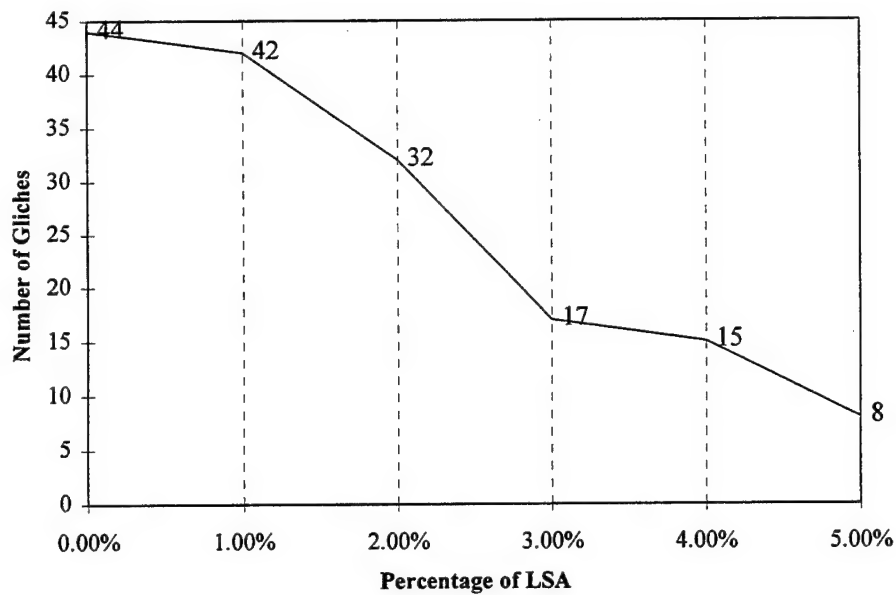


Figure 31. Number of Glitches vs LSA Percentage After LSB-Shift Method

2. Random LSB Method

The random LSB method compares the most significant bits (MSBs) of the odd parity words stopping at the point where the MSBs do not agree. The MSBs that are the same are copied into the MSBs for the even parity word. The LSBs do not come from the original (even parity) word. Instead the LSBs are chosen randomly (0 or 1). The interpolation procedure for three out of five samples having even parity is shown in Figure 32.

The simulation for the random LSB method with the interpolation width changing from 1% to 5% are shown in Figure 33 to Figure 37. The total number of glitches changed randomly, so it is difficult to determine any relationship between the LSA percentage and glitches. The results are summarized in Figure 38 and the randomizing of the LSBs seems to create more problems.

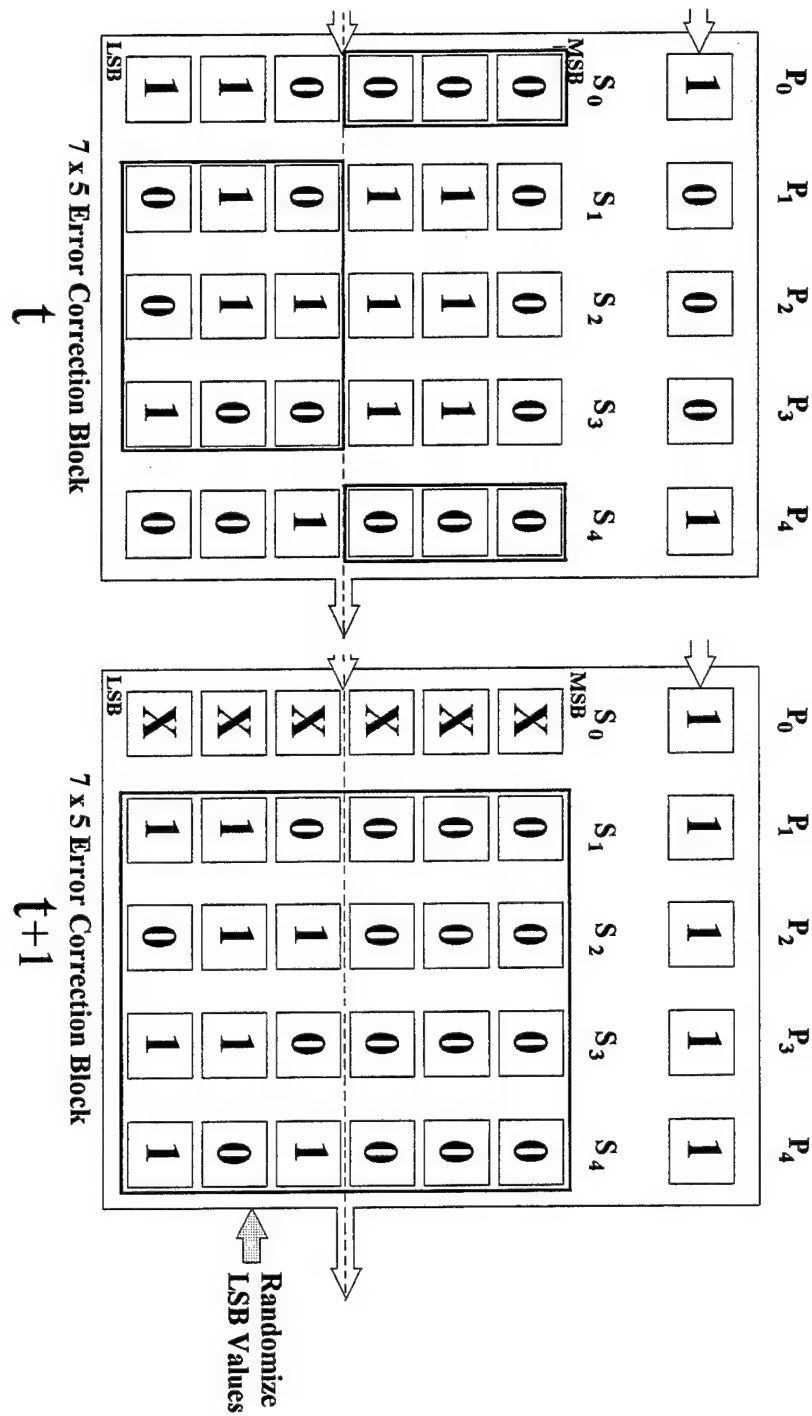


Figure 32. Procedure of Random LSB Method

Interpolation Result, 1 % Random-LSB Method (glitch= 82, sampling distance:ds=0.03 degrees

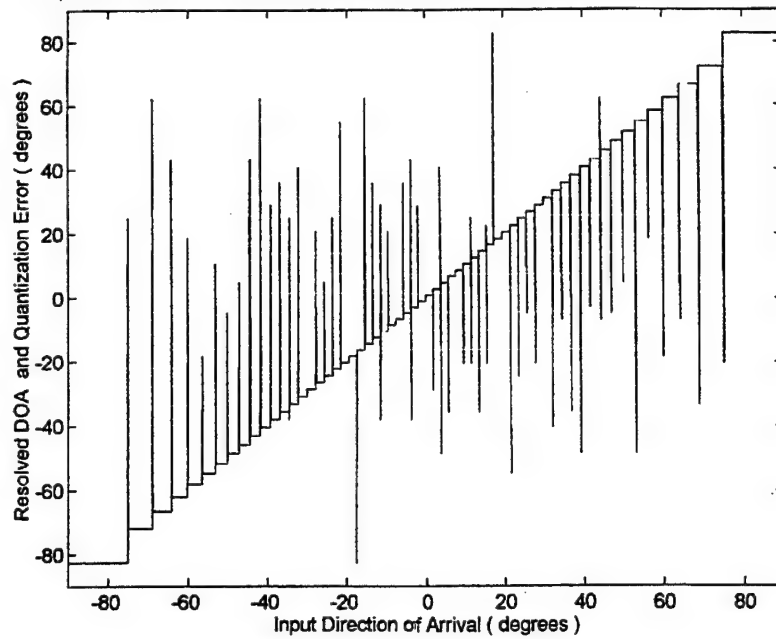


Figure 33. Random-LSB Method Result, 1% of LSA

Interpolation Result, 2 % Random-LSB Method (glitch= 95, sampling distance:ds=0.03 degrees

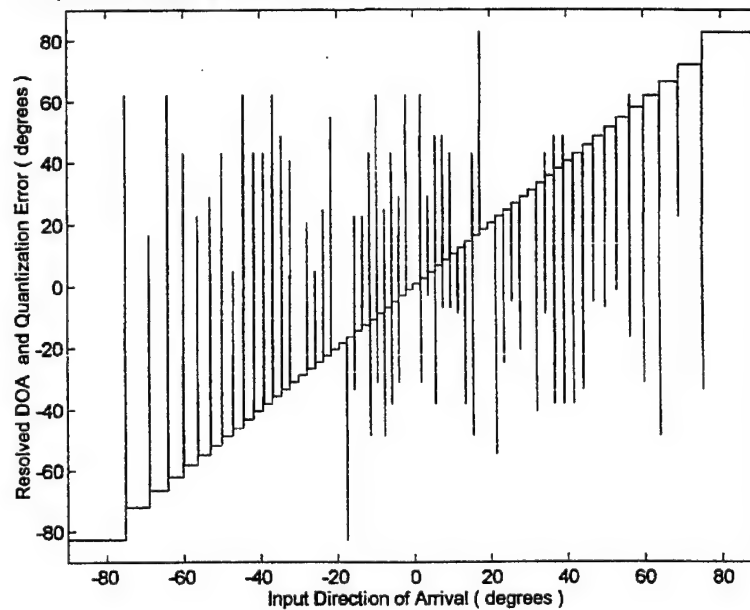


Figure 34. Random-LSB Method Result, 2% of LSA

Interpolation Result, 3 % Random-LSB Method (glitch= 114, sampling distance:ds=0.03 degree)

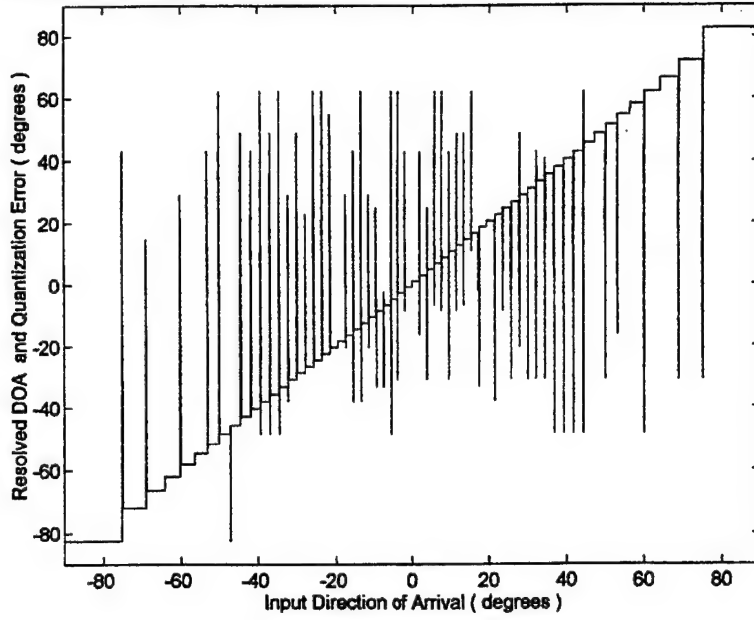


Figure 35. Random-LSB Method Result, 3% of LSA

Interpolation Result, 4 % Random-LSB Method(glitch= 98, sampling distance:ds=0.03 degree)

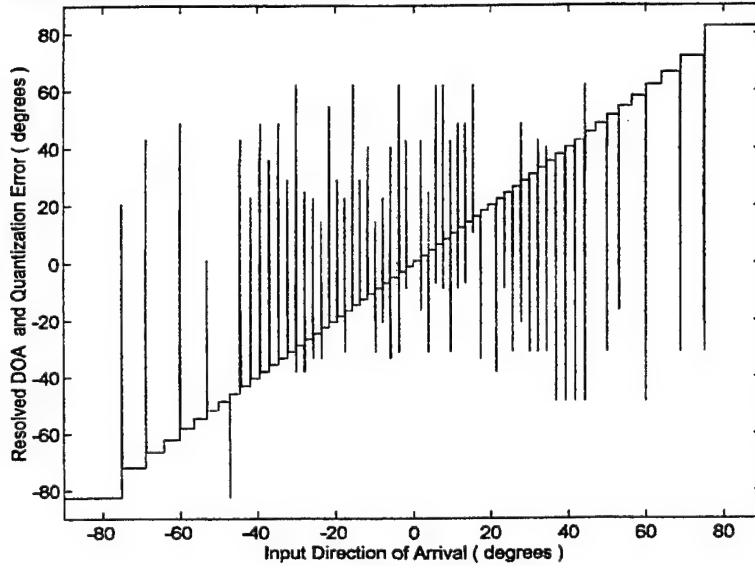


Figure 36. Random-LSB Method Result, 4% of LSA

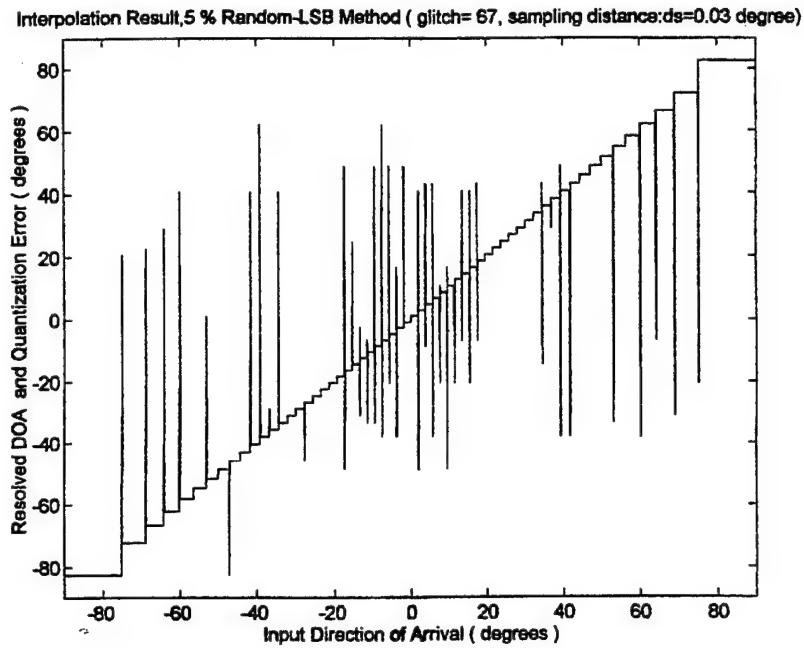


Figure 37. Random-LSB Method Result, 5% of LSA

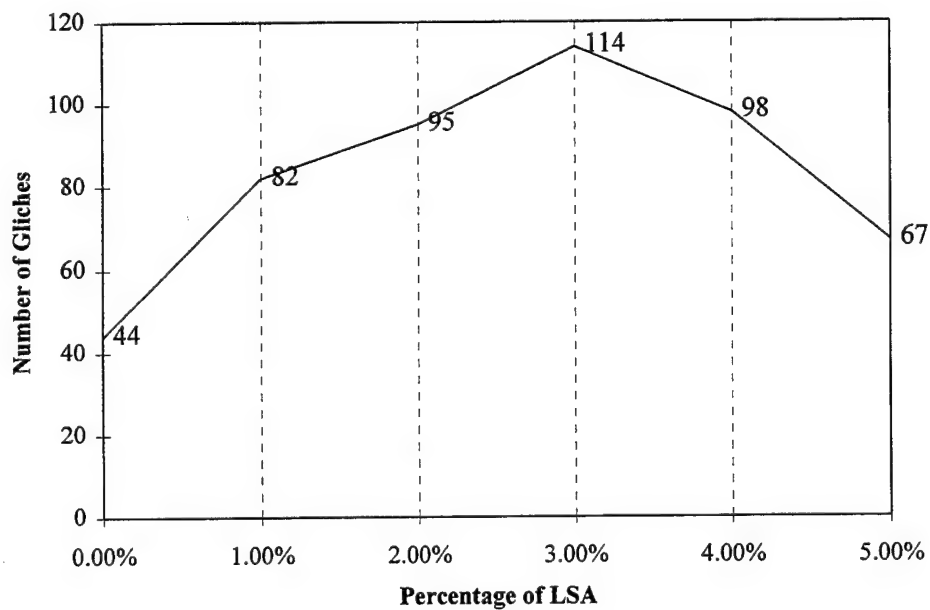


Figure 38. Number of Glitches vs LSA Percentage after Random LSB Method

3. Shift Last Good-Sample Method

In the last two methods, some error samples cannot be resolved completely. This is because some portion of least significant bits remains in the error correction block buffer without changing. With the previous methods some glitches may remain in the sample. However, the shift last good-sample method removes the encoding error perfectly after interpolation by shifting directly from good sample's 6-bit value to error sample's 6-bit value. The procedure of this method is shown in Figure 39.

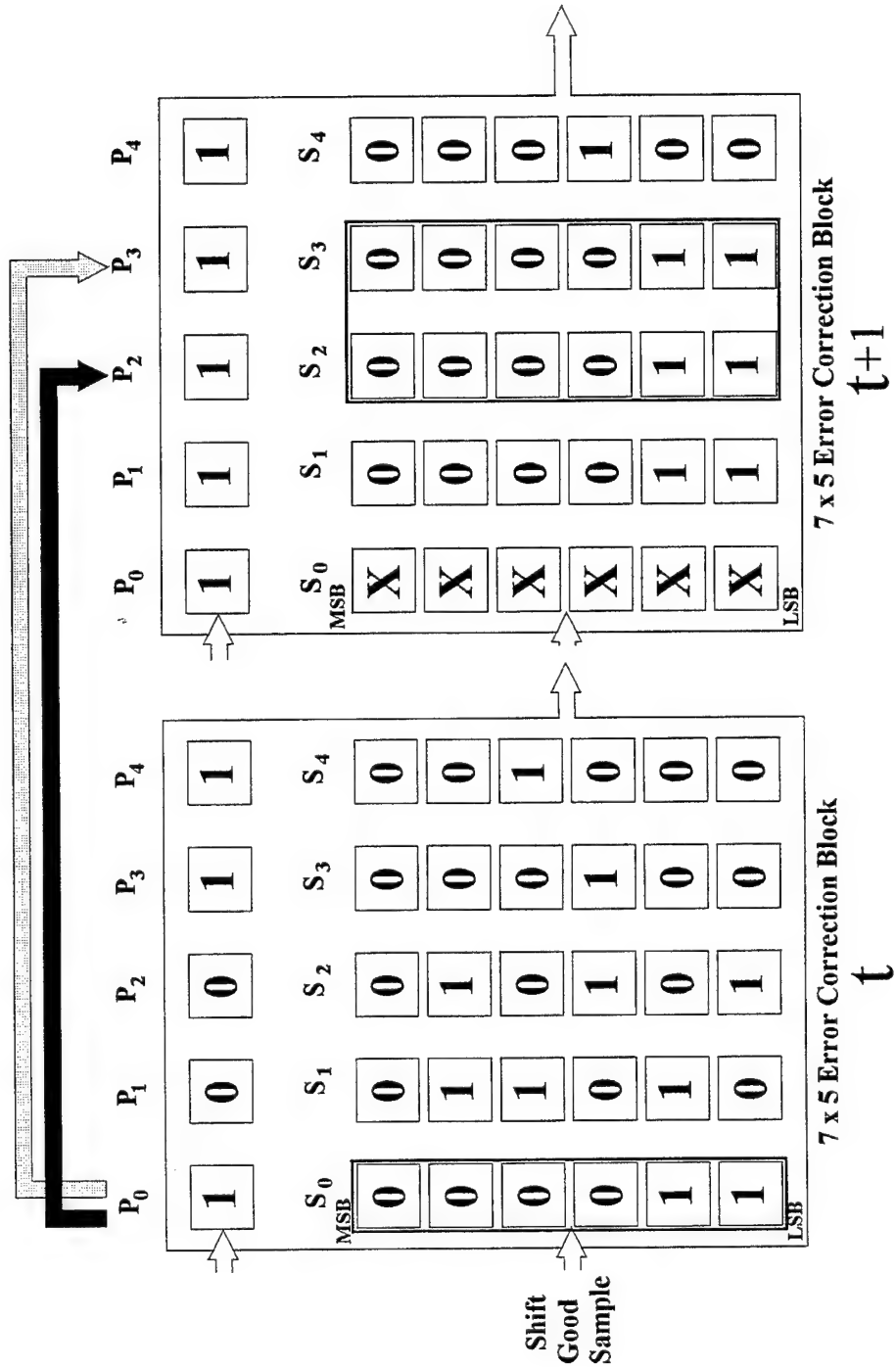


Figure 39. Procedure of Shift Last Good-Sample Method

Interpolation Result, 1 % Shift Last-Good Bit Method (glitch= 36, sampling distance:ds=0.03 degrees)

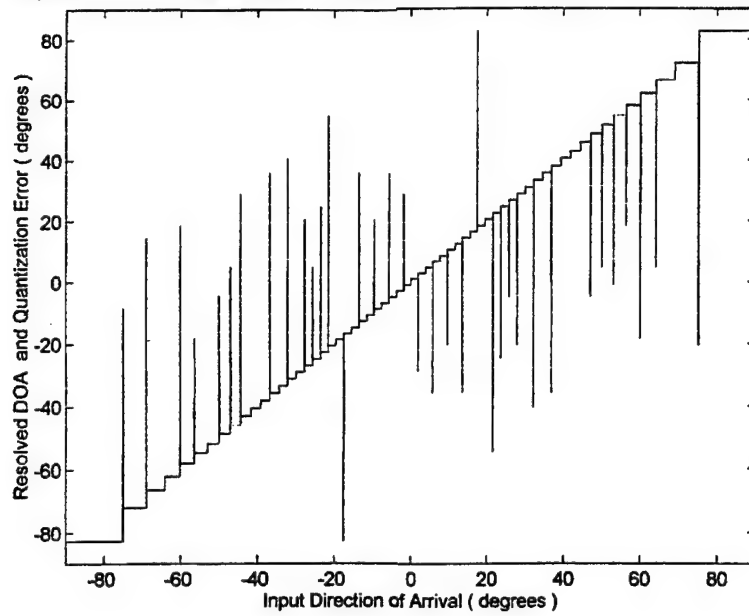


Figure 40. Shift Last Good-Sample Method Result, 1% of LSA

Interpolation Result, 2 % Shift Last-Good Bit Method (glitch= 23, sampling distance:ds=0.03 degrees)

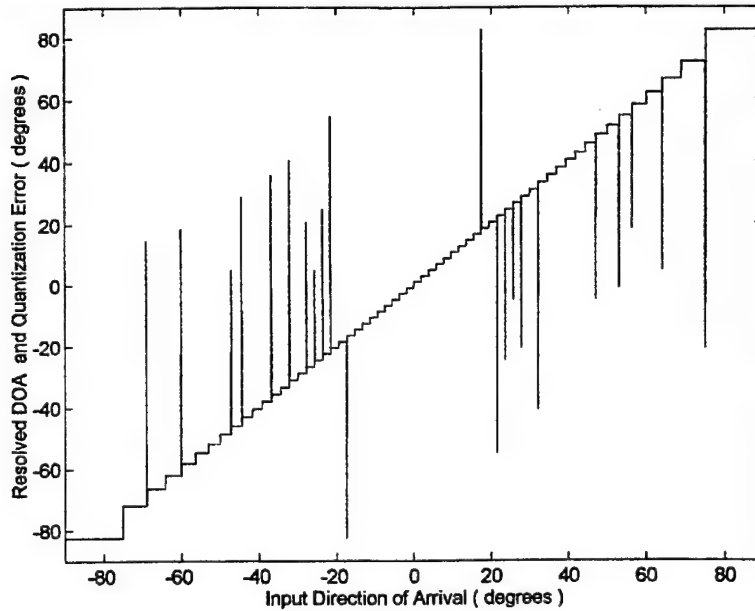


Figure 41. Shift Last Good-Sample Method Result, 2% of LSA

Interpolation Result, 3 % Shift Last-Good Bit Method (glitch= 9, sampling distance:ds=0.03 degrees)

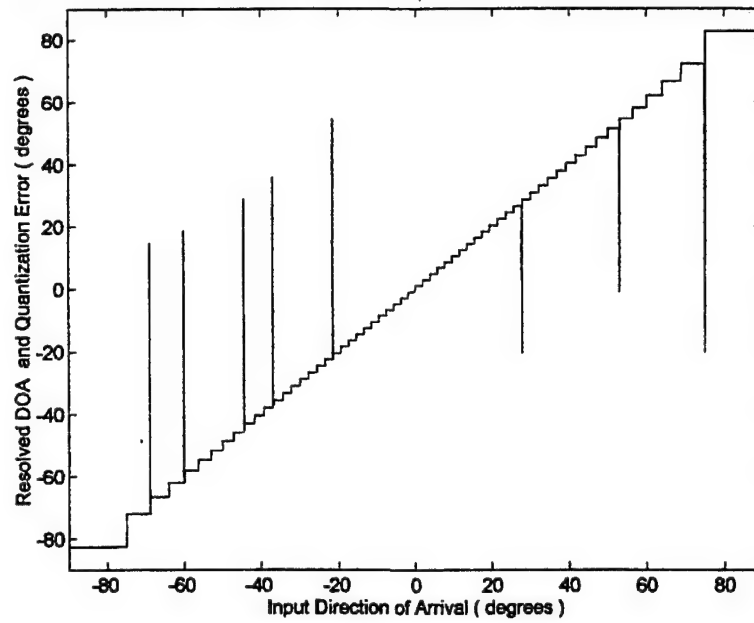


Figure 42. Shift Last Good-Sample Method Result, 3% of LSA

Interpolation Result, 4 % Shift Last-Good Bit Method (glitch= 7, sampling distance:ds=0.03 degrees)

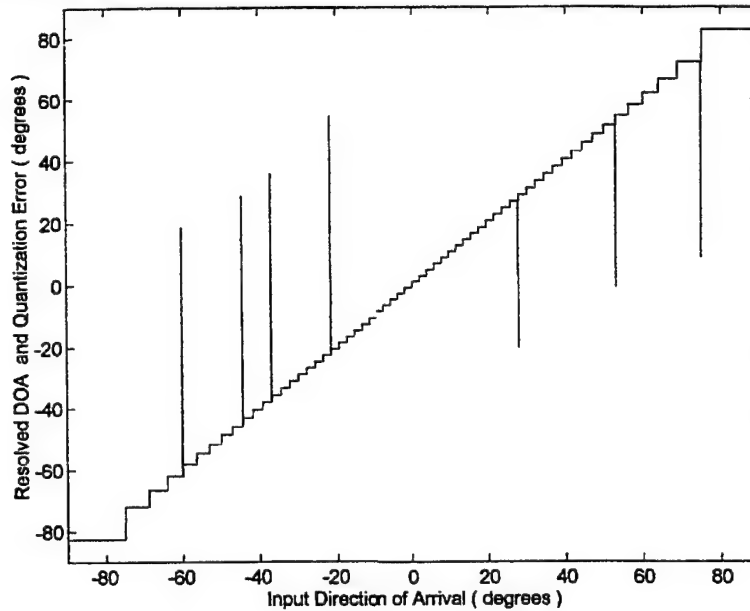


Figure 43. Shift Last Good-Sample Method Result, 4% of LSA

Interpolation Result, 5 % Shift Last-Good Bit Method (glitch= 3, sampling distance:ds=0.03 degrees)

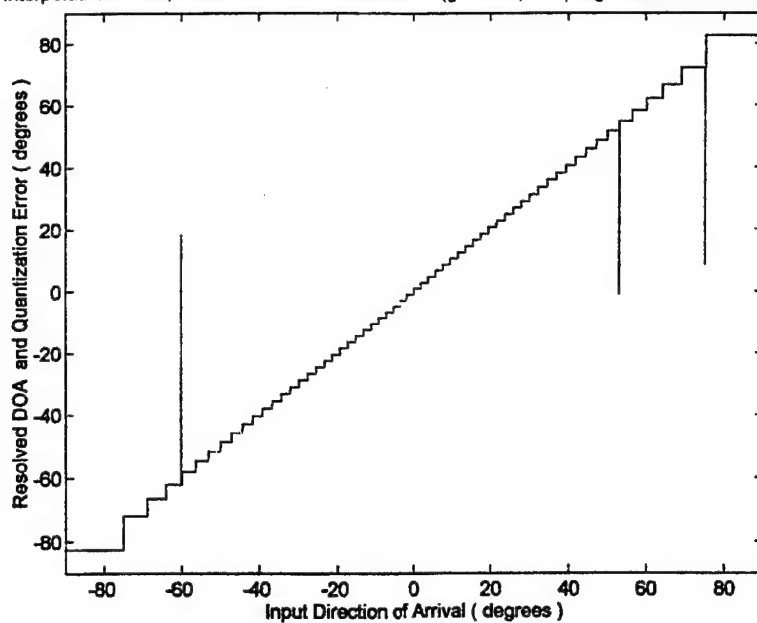


Figure 44. Shift Last Good-Sample Method Result, 5% of LSA

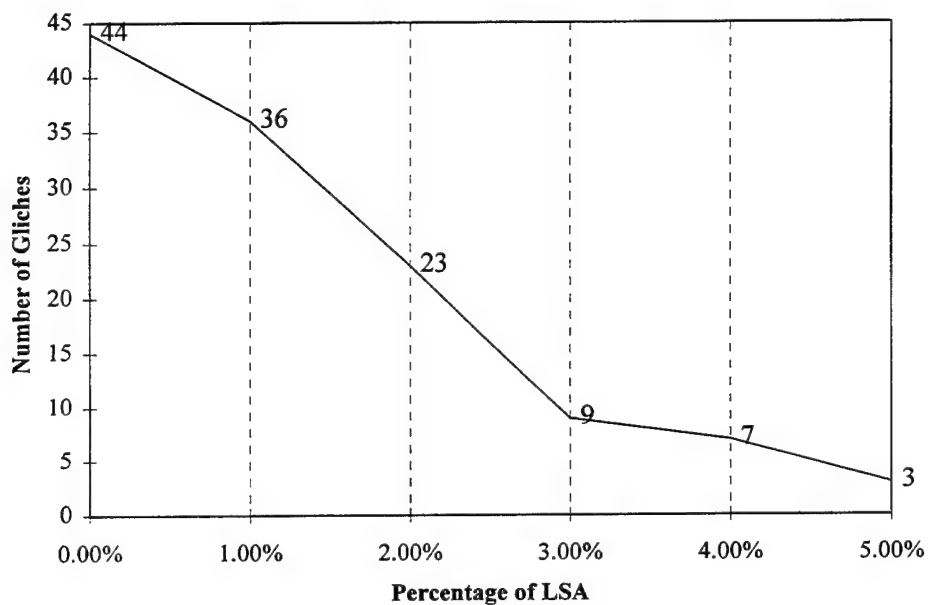


Figure 45. Number of Glitches vs LSA Percentage After Shift Last Good-Sample Method

VI. CONCLUDING REMARKS

The important contribution of this thesis is detailing a residue antenna architecture and interpolation methods that may hold the key to resolving the encoding errors (glitches) using an error correction block. The essential part of interpolation procedure is to determine the additional threshold levels according to LSA percentage in the first modulus ($m_1=3$), so that the parity circuit can be applied to the error correction logic block system. Using the residue number system and the six-bit binary code system it is very fast and develops good resolution for converting the analog input into the direction of arrival (DOA) output.

The three interpolation method results show that the number of glitches is determined by the percentage of LSA in three trials. In the LSB shift method and shift last good sample method, the simulation results indicate that the higher the LSA percentage the better the result (i.e. fewer glitches after interpolation). The random LSB method shows no trends between LSA percentage and the interpolation results because of the number of glitches is randomly changed. A comparison of three interpolation methods is shown in Figure 46.

The simulated results indicate that the RNS provides a more robust solution to high-resolution DF and also has excellent performance. In fact, since the phase response has a "uniform" distribution as a function of angle of arrival it is possible to obtain the direction of arrival information over the entire field of view. To avoid ambiguities in the resolved direction of arrival the spacing between elements must be established for the highest frequency emitter. On the other hand, the frequency of the incoming signal has to be known in order to make the necessary corrections in the RNS logic block using a fast correction algorithm to obtain the correct direction of arrival. The performance of the algorithm was quantified and shown to have excellent results over a large bandwidth. Future work should include a demonstration array with both the microwave and A/D hardware incorporated as shown in Figure 10.

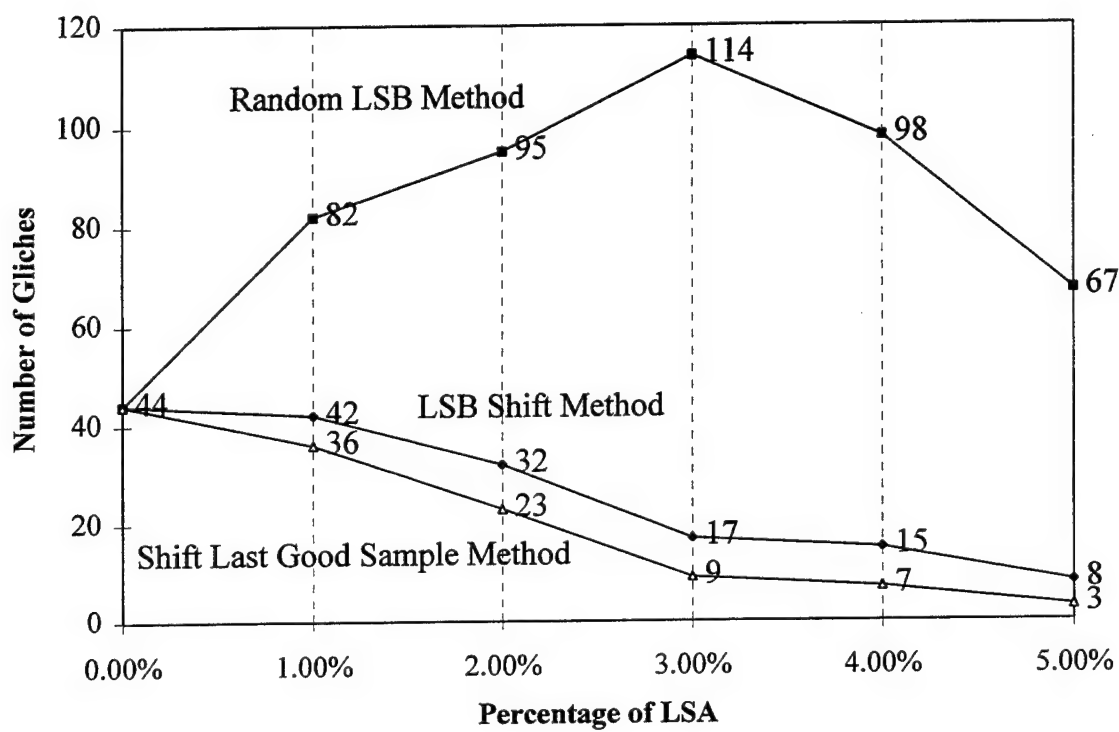


Figure 46. Comparison of the Three Interpolation Method Results

APPENDIX MATLAB CODE

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%      This program 'Phase3' calculates the Direction Of Arrival (DOA) using   %
%      the phase response of the array and RNS code in fast A/D converter      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% program 1
clear all
% Calculate the Dynamic Range
m1=3;
m2=4;
m3=5;
M=m1*m2*m3;
rad=pi/180;

ds=3\input('What is the step size (degree) ? ')
LSA_percent =input('What is the percentage of LSA ?(%)\n')
fo=input('What was the frequency of the antenna (GHz) ?\n')
fr=input('What is the frequency of the received signal(GHz)?\n')
fo=fo*1e9;
fr=fr*1e9;
theta=linspace(-90,90,M/ds);
thet=sin(theta*rad)*90;
c=3e8;
L=c/fo;
L1=c/fr;
% Distances Between Each Pair of Elements
d1=0.5*M/m1*L;
d2=0.5*M/m2*L;
d3=0.5*M/m3*L;

% Generates the Phase Response of Each Pair of Elements
phi1=(2*pi*d1/L1)*sin(theta*rad);
mod1p=angle(exp(j*((phi1+pi)))));
phi2=(2*pi*d2/L1)*sin(theta*rad);
mod2p=angle(exp(j*((phi2)))));
phi3=(2*pi*d3/L1)*sin(theta*rad);
mod3p=angle(exp(j*((phi3)+pi)))));

% Find the Threshold Values for Each Modulus

```

thres1

% Generates the Comparators Output RNS code for Modulus m1=3,m2=4,m3=4

for i=1:M/ds

if mod1p(i)<=T1(1)

mod1(i)=0;

elseif mod1p(i)>T1(2)

mod1(i)=2;

else

mod1(i)=1;

end

if mod2p(i)<=T2(1)

mod2(i)=0;

elseif mod2p(i)<=T2(2)

mod2(i)=1;

elseif mod2p(i)<=T2(3)

mod2(i)=2;

else

mod2(i)=3;

end

if mod3p(i)<=T3(1)

mod3(i)=0;

elseif mod3p(i)<=T3(2)

mod3(i)=1;

elseif mod3p(i)<=T3(3)

mod3(i)=2;

elseif mod3p(i)<=T3(4)

mod3(i)=3;

else

mod3(i)=4;

end

end

figure(1)

whitebg

subplot(611)

plot(theta,mod1p)

axis([-90,90,-4,4]);

title(['Foldingwave and RNS output code (sampling

distance:ds=',num2str(ds*3),'degree')]',' ', num2str(1/ds),'samples in 3 degree)'])

```

grid
subplot(612)
plot(theta,mod2p)
axis([-90,90,-4,4]);
ylabel('Folding waves ')
grid
subplot(613)
plot(theta,mod3p)
axis([-90,90,-4,4]);
grid
subplot(614)
stairs(theta,mod1)
axis([-90,90,-0.2,2.2]);
grid
subplot(615)
stairs(theta,mod2)
axis([-90,90,-0.2,3.2]);
ylabel('RNS Output Code')
grid
subplot(616)
stairs(theta,mod3)
axis([-90,90,-0.2,4.2]);
xlabel('Input Direction of Arrival ( degrees )')
grid

% Find the angle from the 3-bit RNS Modulus code
rnslog1

% Change 3-bit RNS modulus code to six-bit binary code
bin_six

% Find the number of glitch
glitch

% Plots the Resolved DOA and Quantization Error as a Function of the Input Direction of
Arrival
figure(2)
whitebg
plot(theta,th,'b')
xlabel('Input Direction of Arrival ( degrees )')
ylabel('Resolved DOA and Quantization Error ( degrees )')

```

```

title(['Encoding Error in Quatization (glitch= ',num2str(g_count),'sampling
distance:ds=',num2str(ds*3),' degree)'])%, ',num2str(1/ds),'samples in 3 degree ')])
axis([-90 90 -90 90])
grid
print

figure(3)
whitebg
subplot(211)
plot(theta,mod1p)
axis([-90,90,-4,4]);
xlabel('Input Direction of Arrival ( degrees )')
title(['Mode 3 Foldingwave and Output code (sampling distance:ds=',num2str(ds*3),'
degree)'])%, ',num2str(1/ds),'samples in 3 degree ')])
grid

subplot(212)
stairs(theta,mod1)
axis([-90,90,-0.2,2.2]);
xlabel('Input Direction of Arrival ( degrees )')
grid

save bin1
% interpolate the Encoding error
interpl

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This program 'Treshold' calculates the original threshold levels for setting the RNS %
%           folding wave circuit in the 0.3 degree sampling distance           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

%Program 2
% thres.m

```

```

global m1
global m2
global m3
global fo
global fr
global ds
global M
global L

```

```

% Calculates the Threshold Values for Modulus m1
i=-1:2:1;
T1=(pi/3)*i;
T1=sprintf('%6.3f',T1);
T1=sscanf(T1,'%f');

```

```

% Calculates the Threshold Values for Modulus m2
i=-1:1;
T2=(pi/2)*i;
T2=sprintf('%6.3f',T2);
T2=sscanf(T2,'%f');

```

```

% Calculates the Threshold Values for Modulus m3
i=-3:2:3;
T3=(pi/5)*i;
T3=sprintf('%6.3f',T3);
T3=sscanf(T3,'%f');

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%This program 'RNS_ANGLE' is to determine the resolved Direction Of Arrival from %
%                               the RNS output in the moduli set (3,4,5)                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

% program 3

```

```

% set 3-Bit RNS Code

```

```

res=[mod1' mod2' mod3'];

```

```

% Number of Folds Over the Entire field of View for Modulus m1

```

```

nf=fr/fo*M/m1;

```

```

%Total Number of Quantization Levels Over the Entire Field of View

```

```

nql=nf*m1;

```

```

%Generates the Quantization Level Transistion Angles

```

```

p=0:2/nql:1;

```

```

p=[ -rot90(p(2:size(p,2)))' p];

```

```

trang=1/rad*asin(p);

```

```

%Calculates the Midvalue of Each Quantization Level

```

```

for n=1:size(p,2)-1;

```

```

    midval(n)=(trang(n)+(trang(n+1)-trang(n))/2;

```

```

    qlw(n)=midval(n)-trang(n);

```

```

end

```

```

if rem(size(midval,2),2)==0;

```

```

    a=midval;

```

```

    b=size(a,2)/2;

```

```

end

```

```

if rem(size(midval,2),2)==1;

```

```

    a=midval;

```

```

    b=(size(a,2)+1)/2;

```

```

end

```

```

if size(a,2)~=60;

```

```

    a=[zeros(1,b) a zeros(1,b)];

```

```

end

```

```

    b=size(a,2)/2;

```

```

%Calculates the Resolved Direction of Arrival

```

```

for u=1:M/ds;

```

```

    if res(u,:)==([0,0,0]);

```

```

        th(u)=a(1);

```

```

    elseif res(u,:)==([1,1,1]);

```

```

        th(u)=a(2);

```

```

elseif res(u,:)==([2,2,2]);
th(u)=a(3);
elseif res(u,:)==([0,3,3]);
th(u)=a(4);
elseif res(u,:)==([1,0,4]);
th(u)=a(5);
elseif res(u,:)==([2,1,0]);
th(u)=a(6);
elseif res(u,:)==([0,2,1]);
th(u)=a(7);
elseif res(u,:)==([1,3,2]);
th(u)=a(8);
elseif res(u,:)==([2,0,3]);
th(u)=a(9);
elseif res(u,:)==([0,1,4]) ;
th(u)=a(10);
elseif res(u,:)==([1,2,0]);
th(u)=a(11);
elseif res(u,:)==([2,3,1]);
th(u)=a(12);
elseif res(u,:)==([0,0,2]) ;
th(u)=a(13);
elseif res(u,:)==([1,1,3]);
th(u)=a(14);
elseif res(u,:)==([2,2,4]) ;
th(u)=a(15);
elseif res(u,:)==([0,3,0]);
th(u)=a(16);
elseif res(u,:)==([1,0,1]);
th(u)=a(17);
elseif res(u,:)==([2,1,2]) ;
th(u)=a(18);
elseif res(u,:)==([0,2,3]);
th(u)=a(19);
elseif res(u,:)==([1,3,4]) ;
th(u)=a(20);
elseif res(u,:)==([2,0,0]);
th(u)=a(21);
elseif res(u,:)==([0,1,1]);
th(u)=a(22);
elseif res(u,:)==([1,2,2]) ;
th(u)=a(23);

```

```

elseif res(u,:)==([2,3,3]);
th(u)=a(24);
elseif res(u,:)==([0,0,4]);
th(u)=a(25);
elseif res(u,:)==([1,1,0]);
th(u)=a(26);
elseif res(u,:)==([2,2,1]);
th(u)=a(27);
elseif res(u,:)==([0,3,2]) ;
th(u)=a(28);
elseif res(u,:)==([1,0,3]);
th(u)=a(29);
elseif res(u,:)==([2,1,4]) ;
th(u)=a(30);
elseif res(u,:)==([0,2,0]);
th(u)=a(31);
elseif res(u,:)==([1,3,1]);
th(u)=a(32);
elseif res(u,:)==([2,0,2]) ;
th(u)=a(33);
elseif res(u,:)==([0,1,3]);
th(u)=a(34);
elseif res(u,:)==([1,2,4]) ;
th(u)=a(35);
elseif res(u,:)==([2,3,0]);
th(u)=a(36);
elseif res(u,:)==([0,0,1]);
th(u)=a(37);
elseif res(u,:)==([1,1,2]) ;
th(u)=a(38);
elseif res(u,:)==([2,2,3]);
th(u)=a(39);
elseif res(u,:)==([0,3,4]) ;
th(u)=a(40);
elseif res(u,:)==([1,0,0]);
th(u)=a(41);
elseif res(u,:)==([2,1,1]);
th(u)=a(42);
elseif res(u,:)==([0,2,2]) ;
th(u)=a(43);
elseif res(u,:)==([1,3,3]);
th(u)=a(44);

```

```

elseif res(u,:)==([2,0,4]);
th(u)=a(45);
elseif res(u,:)==([0,1,0]);
th(u)=a(46);
elseif res(u,:)==([1,2,1]);
th(u)=a(47);
elseif res(u,:)==([2,3,2]) ;
th(u)=a(48);
elseif res(u,:)==([0,0,3]);
th(u)=a(49);
elseif res(u,:)==([1,1,4]) ;
th(u)=a(50);
elseif res(u,:)==([2,2,0]);
th(u)=a(51);
elseif res(u,:)==([0,3,1]);
th(u)=a(52);
elseif res(u,:)==([1,0,2]) ;
th(u)=a(53);
elseif res(u,:)==([2,1,3]);
th(u)=a(54);
elseif res(u,:)==([0,2,4]) ;
th(u)=a(55);
elseif res(u,:)==([1,3,0]);
th(u)=a(56);
elseif res(u,:)==([2,0,1]);
th(u)=a(57);
elseif res(u,:)==([0,1,2]) ;
th(u)=a(58);
elseif res(u,:)==([1,2,3]);
th(u)=a(59);
else res(u,:)==([2,3,4]) ;
th(u)=a(60);
end
end

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           This program 'SIX_BINARY' converts the RNS code output      %
%           to 6-bit binary code at the instant input angle             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

% program 4
% bin_six.m
% make six-binary code
for u=1:M/ds
    if res(u,:)==([0,0,0])
        th1(u)=1;
        binary(u,:)=( [0 0 0 0 0 0] );

    elseif res(u,:)==([1,1,1])
        th1(u)=2;
        binary(u,:)=( [0 0 0 0 0 1] );

    elseif res(u,:)==([2,2,2])
        th1(u)=3;
        binary(u,:)=( [0 0 0 0 1 0] );

    elseif res(u,:)==([0,3,3])
        th1(u)=4;
        binary(u,:)=( [0 0 0 0 1 1] );

    elseif res(u,:)==([1,0,4])
        th1(u)=5;
        binary(u,:)=( [0 0 0 1 0 0] );

    elseif res(u,:)==([2,1,0])
        th1(u)=6;
        binary(u,:)=( [0 0 0 1 0 1] );

    elseif res(u,:)==([0,2,1])
        th1(u)=7;
        binary(u,:)=( [0 0 0 1 1 0] );

    elseif res(u,:)==([1,3,2])
        th1(u)=8;
        binary(u,:)=( [0 0 0 1 1 1] );

    elseif res(u,:)==([2,0,3])

```

```

th1(u)=9;
binary(u,:)=[0 0 1 0 0 0];

elseif res(u,:)=[0,1,4]
th1(u)=10;
binary(u,:)=[0 0 1 0 0 1];

elseif res(u,:)=[1,2,0]
th1(u)=11;
binary(u,:)=[0 0 1 0 1 0];

elseif res(u,:)=[2,3,1]
th1(u)=12;
binary(u,:)=[0 0 1 0 1 1];

elseif res(u,:)=[0,0,2]
th1(u)=13;
binary(u,:)=[0 0 1 1 0 0];

elseif res(u,:)=[1,1,3]
th1(u)=14;
binary(u,:)=[0 0 1 1 0 1];

elseif res(u,:)=[2,2,4]
th1(u)=15;
binary(u,:)=[0 0 1 1 1 0];

elseif res(u,:)=[0,3,0]
th1(u)=16;
binary(u,:)=[0 0 1 1 1 1];

elseif res(u,:)=[1,0,1]
th1(u)=17;
binary(u,:)=[0 1 0 0 0 0];

elseif res(u,:)=[2,1,2]
th1(u)=18;
binary(u,:)=[0 1 0 0 0 1];

elseif res(u,:)=[0,2,3]
th1(u)=19;
binary(u,:)=[0 1 0 0 1 0];

```

```
elseif res(u,:)==([1,3,4])  
th1(u)=20;  
binary(u,:)=( [0 1 0 0 1 1]);
```

```
elseif res(u,:)==([2,0,0])  
th1(u)=21;  
binary(u,:)=( [0 1 0 1 0 0]);
```

```
elseif res(u,:)==([0,1,1])  
th1(u)=22;  
binary(u,:)=( [0 1 0 1 0 1]);
```

```
elseif res(u,:)==([1,2,2])  
th1(u)=23;  
binary(u,:)=( [0 1 0 1 1 0]);
```

```
elseif res(u,:)==([2,3,3])  
th1(u)=24;  
binary(u,:)=( [0 1 0 1 1 1]);
```

```
elseif res(u,:)==([0,0,4])  
th1(u)=25;  
binary(u,:)=( [0 1 1 0 0 0]);
```

```
elseif res(u,:)==([1,1,0])  
th1(u)=26;  
binary(u,:)=( [0 1 1 0 0 1]);
```

```
elseif res(u,:)==([2,2,1])  
th1(u)=27;  
binary(u,:)=( [0 1 1 0 1 0]);
```

```
elseif res(u,:)==([0,3,2])  
th1(u)=28;  
binary(u,:)=( [0 1 1 0 1 1]);
```

```
elseif res(u,:)==([1,0,3])  
th1(u)=29;  
binary(u,:)=( [0 1 1 1 0 0]);
```

```
elseif res(u,:)==([2,1,4])
```

```

th1(u)=30;
binary(u,:)=( [0 1 1 1 0 1] );

elseif res(u,:)=( [0,2,0] )
th1(u)=31;
binary(u,:)=( [0 1 1 1 1 0] );

elseif res(u,:)=( [1,3,1] )
th1(u)=32;
binary(u,:)=( [0 1 1 1 1 1] );

elseif res(u,:)=( [2,0,2] )
th1(u)=33;
binary(u,:)=( [1 0 0 0 0 0] );

elseif res(u,:)=( [0,1,3] )
th1(u)=34;
binary(u,:)=( [1 0 0 0 0 1] );

elseif res(u,:)=( [1,2,4] )
th1(u)=35;
binary(u,:)=( [1 0 0 0 1 0] );

elseif res(u,:)=( [2,3,0] )
th1(u)=36;
binary(u,:)=( [1 0 0 0 1 1] );

elseif res(u,:)=( [0,0,1] )
th1(u)=37;
binary(u,:)=( [1 0 0 1 0 0] );

elseif res(u,:)=( [1,1,2] )
th1(u)=38;
binary(u,:)=( [1 0 0 1 0 1] );

elseif res(u,:)=( [2,2,3] )
th1(u)=39;
binary(u,:)=( [1 0 0 1 1 0] );

elseif res(u,:)=( [0,3,4] )
th1(u)=40;
binary(u,:)=( [1 0 0 1 1 1] );

```

```
elseif res(u,:)==([1,0,0])  
th1(u)=41;  
binary(u,:)=( [1 0 1 0 0 0]);
```

```
elseif res(u,:)==([2,1,1])  
th1(u)=42;  
binary(u,:)=( [1 0 1 0 0 1]);
```

```
elseif res(u,:)==([0,2,2])  
th1(u)=43;  
binary(u,:)=( [1 0 1 0 1 0]);
```

```
elseif res(u,:)==([1,3,3])  
th1(u)=44;  
binary(u,:)=( [1 0 1 0 1 1]);
```

```
elseif res(u,:)==([2,0,4])  
th1(u)=45;  
binary(u,:)=( [1 0 1 1 0 0]);
```

```
elseif res(u,:)==([0,1,0])  
th1(u)=46;  
binary(u,:)=( [1 0 1 1 0 1]);
```

```
elseif res(u,:)==([1,2,1])  
th1(u)=47;  
binary(u,:)=( [1 0 1 1 1 0]);
```

```
elseif res(u,:)==([2,3,2])  
th1(u)=48;  
binary(u,:)=( [1 0 1 1 1 1]);
```

```
elseif res(u,:)==([0,0,3])  
th1(u)=49;  
binary(u,:)=( [1 1 0 0 0 0]);
```

```
elseif res(u,:)==([1,1,4])  
th1(u)=50;  
binary(u,:)=( [1 1 0 0 0 1]);
```

```
elseif res(u,:)==([2,2,0])
```

```

th1(u)=51;
binary(u,:)=(1 1 0 0 1 0);

elseif res(u,:)=(0,3,1)
th1(u)=52;
binary(u,:)=(1 1 0 0 1 1);

elseif res(u,:)=(1,0,2)
th1(u)=53;
binary(u,:)=(1 1 0 1 0 0);

elseif res(u,:)=(2,1,3)
th1(u)=54;
binary(u,:)=(1 1 0 1 0 1);

elseif res(u,:)=(0,2,4)
th1(u)=55;
binary(u,:)=(1 1 0 1 1 0);

elseif res(u,:)=(1,3,0)
th1(u)=56;
binary(u,:)=(1 1 0 1 1 1);

elseif res(u,:)=(2,0,1)
th1(u)=57;
binary(u,:)=(1 1 1 0 0 0);

elseif res(u,:)=(0,1,2)
th1(u)=58;
binary(u,:)=(1 1 1 0 0 1);

elseif res(u,:)=(1,2,3)
th1(u)=59;
binary(u,:)=(1 1 1 0 1 0);

elseif res(u,:)=(2,3,4)
th1(u)=60;
binary(u,:)=(1 1 1 0 1 1);
end
end
%binary'

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   This program 'Shift_LSB' performs the interpolation using Shift LSB Method   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

% program 5

```

```

% interpl.m

```

```

% set the six-addittional threshol

```

```

adth3

```

```

% This is the operating loop until 'M/ds' number of parity samples

```

```

i=1;

```

```

while i<M/ds

```

```

    a=1; % initial value of 'a' is 1

```

```

        % number of continuous error parity in 5 parity samples

```

```

        % for example, when error is 1, a=2

```

```

        while parity(i)~=1 % to find the first '1' parity(error parity)

```

```

            i=i+1; % i is the location of error

```

```

        end

```

```

        while parity(i)~=0 % to find the next error parity('0' value)

```

```

            i=i+1;

```

```

        end

```

```

        while parity(i)~=1 % to find the continuous number of error parity

```

```

            i=i+1;

```

```

            a=a+1;

```

```

            if i>=M/ds+1,break,end % if the number of parity is

```

```

                % bigger than M/ds,then finish loop !

```

```

        end

```

```

        if i>=M/ds+1,break,end

```

```

        i=i-a+1; % real location value of '0' parity

```

```

        b=1; % highest level of LSB value

```

```

        while binary(i-1,b)==binary(i+a-1,b)

```

```

            b=b+1;

```

```

        if b==7

```

```

            break

```

```

        end

```

```

        end

```

```

    if b==1

```

```

        for k=i+3:-1:i

```

```

            for j=1:6

```

```

                binary(k,j)=binary(k-1,j);

```

```

            end

```

```

        end

```

```

end
% when the number of parity error is one (a=2)
if a==2
    % change the MSB value
    for j=1:b-1
        binary(i+3,j)=binary(i+2,j);
        binary(i+2,j)=binary(i+1,j);
        binary(i+1,j)=binary(i,j);
        binary(i,j)=binary(i-1,j);
    end

    % change the LSB value
    if b<7
        for k=i+3:-1:i
            for j=b:6
                binary(k,j)=binary(k-1,j);
            end
        end
    end

% when the number of parity error is two (a=3)
elseif a==3
    % change the MSB value
    for j=1:b-1
        binary(i+3,j)=binary(i+2,j);
        binary(i+2,j)=binary(i-1,j);
        binary(i+1,j)=binary(i,j);
        binary(i,j)=binary(i-1,j);
    end

    % change the LSB value
    if b<7
        for k=i+3:-1:i
            for j=b:6
                binary(k,j)=binary(k-1,j);
            end
        end
    end

% when the number of parity error is three (a=4)
elseif a==4
    % change the MSB value
    for j=1:b-1

```

```

        binary(i+3,j)=binary(i-1,j);
        binary(i+2,j)=binary(i-1,j);
        binary(i+1,j)=binary(i-1,j);
        binary( i, j)=binary(i-1,j);
    end
    % change the LSB value
    if b<7
        for k=i+3:-1:i
            for j=b:6
                binary(k,j)=binary(k-1,j);
            end
        end
    end
end

% when the number of parity error is four
% a=5
% change the all binary value
else
    for k=i+3:-1:i+1
        for j=1:6
            binary(k,j)=binary(k-1,j);
        end
    end
end % if a==2
% printout 6-bit binary-code after interpolation
if (i+3) < (M/ds)
    v=i-1:i+3;
    parity(v)
    binary(v,:)
end
i=i+a;
end % while i<=M/ds

% change 6-bit binary code to Modulus (m1,m2,m3)
ant_bin

% Calculates the Resolved Direction of Arrival after interpolation
rmslog2

% find the number of glitch
gi_count=0;
for u=1:M/ds-1

```

```

    max(u+1)=th2(u+1)-th2(u);
if u<M/(2*ds)
    if max(u+1)> 11
        parity(u+1)=0;    % possible bad samples
        gi_count=gi_count+1;
        u+1,max(u+1)
        parity(u:u+4)
        binary(u:u+4,:)'
    else
        parity(u+1)=1;    % sample is good
    end
else
    if max(u+1)> 11
        parity(u)=0;    % possible bad samples
        gi_count=gi_count+1;
        u+1,max(u+1)
        parity(u-1:u+3)
        binary(u-1:u+3,:)'
    else
        parity(u)=1;    % sample is good
    end
end % if u<M/(2*ds)
end % for

```

```

figure(5)
whitebg
plot(theta,th2,'b')
xlabel('Input Direction of Arrival ( degrees )')
ylabel('Resolved DOA and Quantization Error ( degrees )')
title(['Interpolation Result,',num2str(LSB_percent),' % LSB-Shift Method ( glitch='
',num2str(gi_count),', sampling distance:ds=',num2str(ds*3),', degree)'])
axis([-90 90 -90 90])
save bin2 T1_add gi_count binary parity;

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%      This program 'ADD_THRESH' calculates the additional threshold levels    %
%      to remove the quantization error                                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

% program 6
% adth3.m

```

```

global m1
global m2
global m3
global fo
global fr
global ds
global rad
global M
global theta
global thet

```

```

%LSB_percent =input('What is the percentage of LSB ?(%)\n')
incre=0.6*LSB_percent/100; % Set the threshold by 1 degree

```

```

% set the additional thresholds
T1_add(1)=2*pi*incre/9-pi;
T1_add(2)=2*pi*(3-incre)/9-pi;
T1_add(3)=2*pi*(3+incre)/9-pi;
T1_add(4)=2*pi*(6-incre)/9-pi;
T1_add(5)=2*pi*(6+incre)/9-pi;
T1_add(6)=2*pi*(9-incre)/9-pi;
T1_add

```

```

% calculate the six-comparator output
for i=1:M/ds
    if mod1p(i) <= T1_add(1)
        count(i) = 0;    % Bad sample
    elseif mod1p(i) <= T1_add(2)
        count(i) = 1;    % Good sample
    elseif mod1p(i) <= T1_add(3)
        count(i) = 2;
    elseif mod1p(i) <= T1_add(4)
        count(i) = 3;
    elseif mod1p(i) <= T1_add(5)

```

```

        count(i) = 4;
    elseif mod1p(i) <= T1_add(6)
        count(i) = 5;
    else
        count(i) = 6;
    end
end
end

```

```

% parity cicuit
for i=1:M/ds
    if rem(count(i),2) == 0
        parity(i)=0;    % possible bad
    else
        parity(i)=1;    % good parity
    end
end
end

```

```

figure(4)
whitebg
subplot(211)
plot(theta,mod1p)
axis([-90,90,-4,4]);
xlabel('Input Direction of Arrival ( degrees )')
grid
subplot(212)
stairs(thet,count)
axis([-90,90,-0.2,6.2]);
xlabel('Input Direction of Arrival ( degrees )')
grid
print

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%      This program 'ANT_BINARY' converts the six-bit binary code      %
%      to the RNS code after interpolation the encoding error          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

% program 7
% ant_bin.m
% change six-binary code to RNS Code
for u=1:M/ds
    if binary(u,:)==([0 0 0 0 0])
        res2(u,:)=(0,0,0);
        th1(u)=1;
    elseif binary(u,:)==([0,0,0,0,0,1])
        res2(u,:)=(1,1,1);
        th1(u)=2;
    %binary(u,:)
    %res(u,:)
    elseif binary(u,:)==([0 0 0 0 1 0])
        res2(u,:)=(2,2,2);
        th1(u)=3;
    elseif binary(u,:)==([0 0 0 0 1 1])
        res2(u,:)=(0,3,3);
        th1(u)=4;
    elseif binary(u,:)==([0 0 0 1 0 0])
        res2(u,:)=(1,0,4);
        th1(u)=5;
    elseif binary(u,:)==([0 0 0 1 0 1])
        res2(u,:)=(2,1,0);
        th1(u)=6;
    elseif binary(u,:)==([0 0 0 1 1 0])
        res2(u,:)=(0,2,1);
        th1(u)=7;
    elseif binary(u,:)==([0 0 0 1 1 1]);
        res2(u,:)=(1,3,2);
        th1(u)=8;
    elseif binary(u,:)==([0 0 1 0 0 0]);
        res2(u,:)=(2,0,3);
        th1(u)=9;
    elseif binary(u,:)==([0 0 1 0 0 1]);
        res2(u,:)=(0,1,4) ;
        th1(u)=10;
    elseif binary(u,:)==([0 0 1 0 1 0]);

```

```

res2(u,:)=[1,2,0];
th1(u)=11;
elseif binary(u,:)=[0 0 1 0 1 1];
res2(u,:)=[2,3,1];
th1(u)=12;
elseif binary(u,:)=[0 0 1 1 0 0];
res2(u,:)=[0,0,2] ;
th1(u)=13;
elseif binary(u,:)=[0 0 1 1 0 1];
res2(u,:)=[1,1,3];
th1(u)=14;
elseif binary(u,:)=[0 0 1 1 1 0];
res2(u,:)=[2,2,4] ;
th1(u)=15;
elseif binary(u,:)=[0 0 1 1 1 1];
res2(u,:)=[0,3,0];
th1(u)=16;
elseif binary(u,:)=[0 1 0 0 0 0];
res2(u,:)=[1,0,1];
th1(u)=17;
elseif binary(u,:)=[0 1 0 0 0 1];
res2(u,:)=[2,1,2] ;
th1(u)=18;
elseif binary(u,:)=[0 1 0 0 1 0];
res2(u,:)=[0,2,3];
th1(u)=19;
elseif binary(u,:)=[0 1 0 0 1 1];
res2(u,:)=[1,3,4] ;
th1(u)=20;
elseif binary(u,:)=[0 1 0 1 0 0];
res2(u,:)=[2,0,0];
th1(u)=21;
elseif binary(u,:)=[0 1 0 1 0 1];
res2(u,:)=[0,1,1];
th1(u)=22;
elseif binary(u,:)=[0 1 0 1 1 0];
res2(u,:)=[1,2,2] ;
th1(u)=23;
elseif binary(u,:)=[0 1 0 1 1 1];
res2(u,:)=[2,3,3];
th1(u)=24;
elseif binary(u,:)=[0 1 1 0 0 0];

```

```

res2(u,:)=(0,0,4);
th1(u)=25;
elseif binary(u,:)=(0 1 1 0 0 1);
res2(u,:)=(1,1,0);
th1(u)=26;
elseif binary(u,:)=(0 1 1 0 1 0);
res2(u,:)=(2,2,1);
th1(u)=27;
elseif binary(u,:)=(0 1 1 0 1 1);
res2(u,:)=(0,3,2);
th1(u)=28;
elseif binary(u,:)=(0 1 1 1 0 0);
res2(u,:)=(1,0,3);
th1(u)=29;
elseif binary(u,:)=(0 1 1 1 0 1);
res2(u,:)=(2,1,4);
th1(u)=30;
elseif binary(u,:)=(0 1 1 1 1 0);
res2(u,:)=(0,2,0);
th1(u)=31;
elseif binary(u,:)=(0 1 1 1 1 1);
res2(u,:)=(1,3,1);
th1(u)=32;
elseif binary(u,:)=(1 0 0 0 0 0);
res2(u,:)=(2,0,2);
th1(u)=33;
elseif binary(u,:)=(1 0 0 0 0 1);
res2(u,:)=(0,1,3);
th1(u)=34;
elseif binary(u,:)=(1 0 0 0 1 0);
res2(u,:)=(1,2,4);
th1(u)=35;
elseif binary(u,:)=(1 0 0 0 1 1);
res2(u,:)=(2,3,0);
th1(u)=36;
elseif binary(u,:)=(1 0 0 1 0 0);
res2(u,:)=(0,0,1);
th1(u)=37;
elseif binary(u,:)=(1 0 0 1 0 1);
res2(u,:)=(1,1,2);
th1(u)=38;
elseif binary(u,:)=(1 0 0 1 1 0);

```

```

res2(u,:)=[2,2,3];
th1(u)=39;
elseif binary(u,:)=[1 0 0 1 1 1];
res2(u,:)=[0,3,4];
th1(u)=40;
elseif binary(u,:)=[1 0 1 0 0 0];
res2(u,:)=[1,0,0];
th1(u)=41;
elseif binary(u,:)=[1 0 1 0 0 1];
res2(u,:)=[2,1,1];
th1(u)=42;
elseif binary(u,:)=[1 0 1 0 1 0];
res2(u,:)=[0,2,2];
th1(u)=43;
elseif binary(u,:)=[1 0 1 0 1 1];
res2(u,:)=[1,3,3];
th1(u)=44;
elseif binary(u,:)=[1 0 1 1 0 0];
res2(u,:)=[2,0,4];
th1(u)=45;
elseif binary(u,:)=[1 0 1 1 0 1];
res2(u,:)=[0,1,0];
th1(u)=46;
elseif binary(u,:)=[1 0 1 1 1 0];
res2(u,:)=[1,2,1];
th1(u)=47;
elseif binary(u,:)=[1 0 1 1 1 1];
res2(u,:)=[2,3,2];
th1(u)=48;
elseif binary(u,:)=[1 1 0 0 0 0];
res2(u,:)=[0,0,3];
th1(u)=49;
elseif binary(u,:)=[1 1 0 0 0 1];
res2(u,:)=[1,1,4];
th1(u)=50;
elseif binary(u,:)=[1 1 0 0 1 0];
res2(u,:)=[2,2,0];
th1(u)=51;
elseif binary(u,:)=[1 1 0 0 1 1];
res2(u,:)=[0,3,1];
th1(u)=52;
elseif binary(u,:)=[1 1 0 1 0 0];

```

```

res2(u,:)=[1,0,2] ;
th1(u)=53;
elseif binary(u,:)==([1 1 0 1 0 1]);
res2(u,:)=[2,1,3];
th1(u)=54;
elseif binary(u,:)==([1 1 0 1 1 0]);
res2(u,:)=[0,2,4] ;
th1(u)=55;
elseif binary(u,:)==([1 1 0 1 1 1]);
res2(u,:)=[1,3,0];
th1(u)=56;
elseif binary(u,:)==([1 1 1 0 0 0]);
res2(u,:)=[2,0,1];
th1(u)=57;
elseif binary(u,:)==([1 1 1 0 0 1]);
res2(u,:)=[0,1,2] ;
th1(u)=58;
elseif binary(u,:)==([1 1 1 0 1 0]);
res2(u,:)=[1,2,3];
th1(u)=59;
elseif binary(u,:)==([1 1 1 0 1 1]);
res2(u,:)=[2,3,4] ;
th1(u)=60;
end
end

%binary'

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   This program 'RNS_angle2' is to determine the resolved Direction Of Arrival   %
%   the RNS output in the moduli set (3,4,5)                                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

% Program 8
% mslog2.m

```

```

% Number of Folds Over the Entire field of View for Modulus m1
nf=fr/fo*M/m1;
%Total Number of Quantization Levels Over the Entire Field of View
nql=nf*m1;
%Generates the Quantization Level Transistion Angles
p=0:2/nql:1;
p=[ -rot90(p(2:size(p,2)))' p];
trang=1/rad*asin(p);

```

```

%Calculates the Midvalue of Each Quantization Level
for n=1:size(p,2)-1
midval(n)=trang(n)+(trang(n+1)-trang(n))/2;
%qlw(n)=midval(n)-trang(n);
end
if rem(size(midval,2),2)==0
a=midval;
b=size(a,2)/2;
end
if rem(size(midval,2),2)==1
a=midval;
b=(size(a,2)+1)/2;
end
if size(a,2)~=60
a=[zeros(1,b) a zeros(1,b)];
end
b=size(a,2)/2;

```

```

%Calculates the Resolved Direction of Arrival
for u=1:M/ds
if res2(u,:)==([0,0,0]);
th2(u)=a(1);
elseif res2(u,:)==([1,1,1]);
th2(u)=a(2);
elseif res2(u,:)==([2,2,2]);

```

```

th2(u)=a(3);
elseif res2(u,:)==([0,3,3]);
th2(u)=a(4);
elseif res2(u,:)==([1,0,4]);
th2(u)=a(5);
elseif res2(u,:)==([2,1,0]);
th2(u)=a(6);
elseif res2(u,:)==([0,2,1]);
th2(u)=a(7);
elseif res2(u,:)==([1,3,2]);
th2(u)=a(8);
elseif res2(u,:)==([2,0,3]);
th2(u)=a(9);
elseif res2(u,:)==([0,1,4]) ;
th2(u)=a(10);
elseif res2(u,:)==([1,2,0]);
th2(u)=a(11);
elseif res2(u,:)==([2,3,1]);
th2(u)=a(12);
elseif res2(u,:)==([0,0,2]) ;
th2(u)=a(13);
elseif res2(u,:)==([1,1,3]);
th2(u)=a(14);
elseif res2(u,:)==([2,2,4]) ;
th2(u)=a(15);
elseif res2(u,:)==([0,3,0]);
th2(u)=a(16);
elseif res2(u,:)==([1,0,1]);
th2(u)=a(17);
elseif res2(u,:)==([2,1,2]) ;
th2(u)=a(18);
elseif res2(u,:)==([0,2,3]);
th2(u)=a(19);
elseif res2(u,:)==([1,3,4]) ;
th2(u)=a(20);
elseif res2(u,:)==([2,0,0]);
th2(u)=a(21);
elseif res2(u,:)==([0,1,1]);
th2(u)=a(22);
elseif res2(u,:)==([1,2,2]) ;
th2(u)=a(23);
elseif res2(u,:)==([2,3,3]);

```

```

th2(u)=a(24);
elseif res2(u,:)==([0,0,4]);
th2(u)=a(25);
elseif res2(u,:)==([1,1,0]);
th2(u)=a(26);
elseif res2(u,:)==([2,2,1]);
th2(u)=a(27);
elseif res2(u,:)==([0,3,2]) ;
th2(u)=a(28);
elseif res2(u,:)==([1,0,3]);
th2(u)=a(29);
elseif res2(u,:)==([2,1,4]) ;
th2(u)=a(30);
elseif res2(u,:)==([0,2,0]);
th2(u)=a(31);
elseif res2(u,:)==([1,3,1]);
th2(u)=a(32);
elseif res2(u,:)==([2,0,2]) ;
th2(u)=a(33);
elseif res2(u,:)==([0,1,3]);
th2(u)=a(34);
elseif res2(u,:)==([1,2,4]) ;
th2(u)=a(35);
elseif res2(u,:)==([2,3,0]);
th2(u)=a(36);
elseif res2(u,:)==([0,0,1]);
th2(u)=a(37);
elseif res2(u,:)==([1,1,2]) ;
th2(u)=a(38);
elseif res2(u,:)==([2,2,3]);
th2(u)=a(39);
elseif res2(u,:)==([0,3,4]) ;
th2(u)=a(40);
elseif res2(u,:)==([1,0,0]);
th2(u)=a(41);
elseif res2(u,:)==([2,1,1]);
th2(u)=a(42);
elseif res2(u,:)==([0,2,2]) ;
th2(u)=a(43);
elseif res2(u,:)==([1,3,3]);
th2(u)=a(44);
elseif res2(u,:)==([2,0,4]);

```

```

th2(u)=a(45);
elseif res2(u,:)==([0,1,0]);
th2(u)=a(46);
elseif res2(u,:)==([1,2,1]);
th2(u)=a(47);
elseif res2(u,:)==([2,3,2]) ;
th2(u)=a(48);
elseif res2(u,:)==([0,0,3]);
th2(u)=a(49);
elseif res2(u,:)==([1,1,4]) ;
th2(u)=a(50);
elseif res2(u,:)==([2,2,0]);
th2(u)=a(51);
elseif res2(u,:)==([0,3,1]);
th2(u)=a(52);
elseif res2(u,:)==([1,0,2]) ;
th2(u)=a(53);
elseif res2(u,:)==([2,1,3]);
th2(u)=a(54);
elseif res2(u,:)==([0,2,4]) ;
th2(u)=a(55);
elseif res2(u,:)==([1,3,0]);
th2(u)=a(56);
elseif res2(u,:)==([2,0,1]);
th2(u)=a(57);
elseif res2(u,:)==([0,1,2]) ;
th2(u)=a(58);
elseif res2(u,:)==([1,2,3]);
th2(u)=a(59);
else res2(u,:)==([2,3,4]) ;
th2(u)=a(60);
end
end

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This program 'Last_Good' performs the interpolation using Last good-sample shift %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

% program 9

```

```

% int_copy

```

```

% set the six-addittional threshol

```

```

adth3

```

```

% This is the operating loop until 'M/ds' number of parity samples

```

```

i=1;

```

```

while i<M/ds

```

```

    a=1; % initial value of 'a' is 1

```

```

        % number of continuous error parity in 5 parity samples

```

```

        % for example, when error is 1, a=2

```

```

        while parity(i)~=1 % to find the first '1' parity(error parity)

```

```

            i=i+1; % i is the location of error

```

```

        end

```

```

        while parity(i)~=0 % to find the next error parity('0' value)

```

```

            i=i+1;

```

```

        end

```

```

        while parity(i)~=1 % to find the continuous number of error parity

```

```

            i=i+1;

```

```

            a=a+1;

```

```

            if i>=M/ds+1,break,end % if the number of parity is

```

```

                % bigger than M/ds,then finish loop !

```

```

        end

```

```

        if i>=M/ds+1,break,end

```

```

        i=i-a+1; % real location value of '0' parity

```

```

        b=1; % highest level of LSB value

```

```

% Copy the last good sample to error samples

```

```

for k=i+a:-1:i

```

```

    for j=1:6

```

```

        binary(k,j)=binary(i-2,j);

```

```

    end

```

```

end

```

```

i=i+1;

```

```

end % while i<=M/ds

```

```

% change 6-bit binary code to Modulus (m1,m2,m3)

```

```

ant_bin

```

```

% Calculates the Resolved Direction of Arrival after interpolation
rnslog2

% find the number of glitch
%parity=ones(1,M/ds);
gi_count=0;
for u=1:M/ds-1
    max(u+1)=th2(u+1)-th2(u);
    if u<M/(2*ds)
        if max(u+1)> 11
            parity(u+1)=0;    % possible bad samples
            gi_count=gi_count+1;
            u+1,max(u+1)
            parity(u:u+4)
            binary(u:u+4,:)'
        else
            parity(u+1)=1;    % sample is good
        end
    else
        if max(u+1)> 11
            parity(u)=0;    % possible bad samples
            gi_count=gi_count+1;
            u+1,max(u+1)
            parity(u-1:u+3)
            binary(u-1:u+3,:)'
        else
            parity(u)=1;    % sample is good
        end
    end % if u<M/(2*ds)
end % for

figure(5)
whitebg
plot(theta,th2,'b')
xlabel('Input Direction of Arrival ( degrees )')
ylabel('Resolved DOA and Quantization Error ( degrees )')
title(['Interpolation Result,',num2str(LSB_percent),' % Shift Last-Good Bit Method (glitch='
%,num2str(gi_count),' , sampling distance:ds=',num2str(ds*3),' degree)'])
axis([-90 90 -90 90])
print
save bin3 T1_add gi_count binary parity;

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This program 'RAND_LSB' performs the interpolation using random LSB Method
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

%program 10
% int_rand
% set the six-addittional threshol
adth3

```

```

% This is the operating loop until 'M/ds' number of parity samples

```

```

i=1;
while i<M/ds
    a=1; % initial value of 'a' is 1
        % number of continuous error parity in 5 parity samples
        % for example, when error is 1, a=2

    while parity(i)~=1 % to find the first '1' parity(error parity)
        i=i+1; % i is the location of error
    end
    while parity(i)~=0 % to find the next error parity('0' value)
        i=i+1;
    end
    while parity(i)~=1 % to find the continuous number of error parity
        i=i+1;
        a=a+1;
        if i>=M/ds+1,break,end % if the number of parity is
            % bigger than M/ds,then finish loop !
    end
    if i>=M/ds+1,break,end
    i=i-a+1; % real location value of '0' parity
    b=1; % highest level of LSB value
    while binary(i-1,b)==binary(i+a-1,b)
        b=b+1;
        if b==7
            break
        end
    end
    if b==1
        for k=i+3:-1:i
            binary(k,:)=rem(randperm(6),2);
        end
    end
end

```

```

end

% when the number of parity error is one (a=2)
if a==2
    % change the MSB value
    for j=1:b-1
        binary(i+3,j)=binary(i+2,j);
        binary(i+2,j)=binary(i+1,j);
        binary(i+1,j)=binary(i,j);
        binary(i,j)=binary(i-1,j);
    end

    % randomize the LSB value
    if b<7
        for k=i+3:-1:i
            binary(k,:)=rem(randperm(6),2);
        end
    end

% when the number of parity error is two (a=3)
elseif a==3
    % change the MSB value
    for j=1:b-1
        binary(i+3,j)=binary(i+2,j);
        binary(i+2,j)=binary(i-1,j);
        binary(i+1,j)=binary(i-1,j);
        binary(i,j)=binary(i-1,j);
    end

    % change the LSB value
    if b<7
        for k=i+3:-1:i
            for j=b:6
                binary(k,j)=binary(k-1,j);
            end
        end
    end

% when the number of parity error is three (a=4)
elseif a==4
    % change the MSB value
    for j=1:b-1
        binary(i+3,j)=binary(i-1,j);
    end
end

```

```

        binary(i+2,j)=binary(i-1,j);
        binary(i+1,j)=binary(i-1,j);
        binary(i,j)=binary(i-1,j);
    end
    % change the LSB value
    if b<7
        for k=i+3:-1:i
            binary(k,:)=rem(randperm(6),2);
        end
    end

    % when the number of parity error is four
    % a=5
    % change the all binary value
    else
        for k=i+3:-1:i+1
            for j=1:6
                binary(k,j)=binary(k-1,j);
            end
        end
    end
    end % if a==2

    % printout 6-bit binary-code after interpolation
    i
    if (i+3) < (M/ds)
        v=i-1:i+3;
        parity(v)
        binary(v,:)
    end
    i=i+a;
    end % while i<=M/ds

    % change 6-bit binary code to Modulus (m1,m2,m3)
    ant_bin

    % Calculates the Resolved Direction of Arrival after interpolation
    rnslog2

    % find the number of glitch
    parity=ones(1,M/ds);
    gi_count=0;
    for u=1:M/ds-1

```

```

    max(u+1)=th2(u+1)-th2(u);
if u<M/(2*ds)
    if max(u+1)> 11
        parity(u+1)=0;    % possible bad samples
        gi_count=gi_count+1;
        u+1,max(u+1)
        parity(u:u+4)
        binary(u:u+4,:)'
    else
        parity(u+1)=1;    % sample is good
    end
else
    if max(u+1)> 11
        parity(u)=0;    % possible bad samples
        gi_count=gi_count+1;
        u+1,max(u+1)
        parity(u-1:u+3)
        binary(u-1:u+3,:)'
    else
        parity(u)=1;    % sample is good
    end
end % if u<M/(2*ds)
end % for

figure(5)
whitebg
plot(theta,th2,'b')
xlabel('Input Direction of Arrival ( degrees )')
ylabel('Resolved DOA and Quantization Error ( degrees )')
title(['Interpolation Result,',num2str(LSB_percent),' % Random-LSB Method ( glitch=']
%',num2str(gi_count),' , sampling distance:ds=',num2str(ds*3),' degree')])
axis([-90 90 -90 90])
%print

save bin4 T1_add gi_count binary parity;

```

LIST OF REFERENCES

1. Unnikrishna, Pillai, S., "A New Approach to Array Geometry for Improved Spatial Spectrum Estimation," *IEEE*, vol. 73, no. 10, October 1985.
2. Rodrigues, Louis and E. Moita, "High-resolution Residue Antenna Architectures for Wideband Direction Finding," Master Thesis, Naval Postgraduate School, June 1996.
3. Pace, P. E., L. E. M. Rodrigues and D. C. Jenn, "High Resolution Residue Antenna Architectures for Wideband Direction Finding, Department of Electrical and Computer Engineering," submitted for presentation at the 1997 National Radar Conference, June 1996.
4. Travers, D. N., and S. M. Hixon, *Abstracts of Available Literature on Radio Direction Finding*, Southwest Research Institute, 1966.
5. Benoit, R. C. Jr., and F. Coughlin Jr., "Designing RDF Antennas," *Electronic Industries*, vol.18, pp.77-83, April 1959.
6. Jenkins, Hernodon H., *Small Aperture Radio Direction Finding*, Norwood, MA: Artech House Inc., 1991.
7. Simon, Haykin, *Radio Array Signal Processing for Angle of Arrival Estimation*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1985.
8. Grosswald, E., *Topics From the Theory of Numbers*, McMillan, New York, 1966.
9. Gething, P. J. D., "High-Frequency Direction Finding," *Proceedings of the IEEE*, vol. 113, pp. 49-61, January 1966.
10. Rhodes, D. R., *Introduction to Monopulse Radar*, McGraw-Hill, 1959.
11. Frank, Jay, ed., *IEEE Standard Dictionary of Electrical and Electronic Terms ANSI/IEEE Standard*, Institute of Electrical Engineers, Inc., New York, 1988.

INITIAL DISTRIBUTION LIST

	No. Copies
1. Defense Technical Information Center 8725 John J. Kingman Rd., STE 0944 Ft. Belvoir, VA 22060-6218	2
2. Dudley Knox Library Naval Postgraduate School 411 Dyer Rd. Monterey, California 93943-5101	2
3. Chairman, Code EW Department of Electronic Warfare Academic Group Naval Postgraduate School Monterey, California 93943-5121	1
4. Professor David C. Jenn, Code EC/Jn Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943-5121	1
5. Professor Phillip. E. Pace, Code EC/Pc Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943-5121	3
6. Space and Naval Warfare System Command Department of the Navy PMW-163 Attn: CAPT. Connell Washington, DC 20363-5100	1
7. Space and Naval Warfare System Command Department of the Navy PMW-163 Attn: Mike Wilson Washington, DC 20363-5100	1

- | | | |
|-----|---|---|
| 8. | Space and Naval Warfare System Command
Department of the Navy
PMW-163
Attn: Ralph Skiano
Washington, DC 20363-5100 | 1 |
| 9. | Space and Naval Warfare System Command
Department of the Navy
PMW-163
Attn: CAPT. Ristorcelli
Washington, DC 20363-5100 | 1 |
| 10. | Byeong-Jun Park
368-14 Sansu-Dong
Kwangju, 501-091
Republic of Korea | 2 |
| 11. | Library
P.O.box 77, Gongneung-Dong
Dobong-Gu, Seoul, 132-240
Republic of Korea | 1 |
| 12. | Library
Army Headquarters, Bunam-Ri, Duma-Myun
Nonsan-Gun, Chungnam-Do, 320-919
Republic of Korea | 1 |